STRUCTURED VISUAL UNDERSTANDING, GENERATION AND REASONING

A Dissertation
Presented to
The Academic Faculty

By

Jianwei Yang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology

May 2020

STRUCTURED VISUAL UNDERSTANDING, GENERATION AND REASONING

Approved by:

Dr. Devi Parikh, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Dhruv Batra
School of Interactive Computing
Georgia Institute of Technology

Dr. David J. Crandall
School of Informatics, Computing, and Engineering
Indiana University

Dr. Stefan Lee
School of Electrical Engineering and Computer Science
Oregon State University

Dr. Judy Hoffman
School of Interactive Computing
Georgia Institute of Technology

Date Approved: November 20, 2019

ACKNOWLEDGEMENTS

Earning a Ph.D. is never easy. In my past five years of Ph.D. study, there are so many people who guided and helped me. I want to express my greatest thanks to all of them.

First of all, I would like to thank my Ph.D. advisor, Prof. Devi Parikh. I cannot express how grateful I am for her guidance and support. After the failure of Ph.D. program application at 2014, I was really privileged to get the offer from Devi's group at 2015. Since then, Devi is the one who gave me the most guidance and support. For researches, Devi has helped me a lot to set up a good mindset to do a good research and write a good paper. She also provided me enough freedom to explore my own interests. I really appreciate the guidance Devi gave in terms of high-level research philosophy. Besides researches, Devi is also extremely helpful at many other aspects. Her suggestions on how to become a good presenter, how to become a good paper writer and how to become a good time manager, have profoundly affected me and will be definitely beneficial to me for many years in the future. I would also like to thank Prof. Dhruv Batra. I always think Dhruv as my co-advisor due to his supports in my Ph.D. study. Dhruv participated in almost all my research projects and he always gives me very helpful suggestions on problem definition, paper writing, formulation, etc. Though sometimes it is tough to me, his high standard on researches always push me to try my best to make a solid work!

I would also thank my other mentors during my Ph.D. study. I would like to thank Dr. Anitha Kanan for her guidance when I was interning at Facebook AI Research (FAIR) at 2016. Her passion and curiosity on researches was really impressive to me and inspired me a lot during the internship. I would like to thank Dr. Ning Zhang, Dr. Sergey Tulyakov and Dr. Linjie Yang, who are my mentors during my internship at Snap. All of them are very supportive and always have good suggestions to the research project. I would like to thank Dr. Chuang Gan, my mentor at MIT-IBM Watson AI Lab, for his supports on both researches and life. Also thanks for bringing me to Boston. It is a really amazing city. At

last, I would like to thank Prof. Stefan Lee. He is extremely thoughtful and can always give me good suggestions on my researches. I enjoyed much to collaborate with him during my last two years of Ph.D.

I would also like to thank my labmates at Computer Vision and Machine Learning and Perception (CVMLP) lab. I would like to thank Jiasen Lu, who is my most important research partner during my Ph.D. study. Since I first visited CVMLP lab, I met Jiasen and then we became the closest partner. Not only the suggestions on researches, Jiasen also gives a lot of supports in my daily life. Thanks for borrowing me your car so that I can practice my driving skill. Thanks for helping with the bandage when my both bands are burn heavily. Thanks for driving my between apartment and campus when I do not have a car. I would also like to thank Peng Zhang, Xiao Lin and Qing Sun. Peng helped me a lot to get to know about the new environment and gave me many good suggestions on my daily life. I still remembered the Fall semester when I was interning at FAIR with Xiao. He is so thoughtful about researches and can always see the essence behind the noisy observations, which inspired my tremendously afterwards. Qing earns my greatest respect on researches. She has a very high bar on researches and great perseverance to pursuit the research goals. I would also like to thank Ramakrishna Vedantam. It is really enjoyable to discuss with Rama in front of the whiteboard. Rama is so thoughtful and smart. He can always formulate a vague problem into a clean equation and then explain it to me clearly. I was always amazed by his smart and inspired by his insights on many problems. I would also like thank Zhile Ren and Mingze Xu. It is really a great time to collaborate with Zhile and Mingze in my last two years of Ph.D. study. Zhile has a very good expertise in many research areas, especially 3D vision, and can always give some good solutions to the tough problems during our collaborations. Mingze has a lot of research experiences on computer vision and is always supportive and hard-working during our collaborations. Finally, I would like to thank Stanislaw Antol, Yash Goyal, Aishwarya Agarwal, Michael Cogswell, Ramprasaath Selvaraju, Purva Tendulkar, Akrit Mohapatra, Harsh Agarwal, Abhishek Das,

Arjun Chandrasekharan, Ashwin Kalyan, Prithvijit Chattopadhyay, Satwik Kottur, Deshraj Yadav, Viraj Prabhu, Samyak Datta, Nirbhay Modhe, Ayush Shrivastava, Mohit Sharma and Peter Anderson and many others in CVMLP lab.

Finally, I would like to thank my parents and fiancée for their love and support. In the last five years, though far away from home, my father and mother are always the first to reach out and care about my study and life. No matter sweet or bitter, they always encourage me, support me to move forward. Two years ago, I was very fortunate to meet my fiancée Yiyuan Fu. In the last two years, we started from being strangers to each other but now have become the closest one to each other in the world. It is not possible for me to accomplish my Ph.D. study without her inspiration, support and love. I love you forever!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The world around us is highly structured. In the real world, a single object usually consists of multiple components organized in some structures (*e.g.*, a person has different body parts), and multiple objects usually exist in a scene and interact with each other in predictable ways (*e.g.*, man playing basketball). This structure manifests itself in the visual data that captures the world around us and in the text describing it and thus can potentially provide a strong inductive bias to various vision tasks. In this thesis, we focus on exploiting the structures existing in visual data to improve visual understanding, generation and reasoning. Specifically, for visual understanding, we model structure at different levels to improve image classification, scene graph generation and representation learning. In visual generation, we exploit the foreground-background structure in images to generate images in a layer-wise manner to reduce blending artifacts between foreground and background. Finally, we use the structured visual representations as the intermediate interface to bridge visual perception and reasoning to address different vision and language tasks, including image captioning and visual question generation. Through extensive experiments, we demonstrate that leveraging structure in visual data can not only improve the model performance, but also make vision and language models more grounded and interpretable.

CHAPTER 1
INTRODUCTION

Ubiquitous access to digital cameras has led to unprecedentedly large quantities of visual data being recorded for news, entertainment, social media and personal logging. Frequently augmented with text, these images and videos are a dominant medium via which we communicate information and express ourselves.

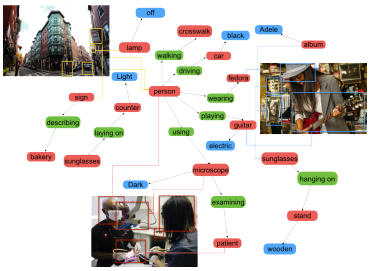
The world around us is highly structured. An object usually consists of multiple semantic components in some specific spatial configuration. A scene usually contains multiple objects which interact with each other in predictable ways. This structure manifests itself in the visual data that captures the world around us, and in text that describes it. Take Fig. 1.1 as an example. In Fig. 1.1 (a), each image contains multiple people and each person can be labeled into a number of body parts, such as arms and legs. Similarly in Fig. 1.1 (b), each image consists of a background and several foreground objects, which probably have some relationships with each other as shown in Fig. 1.1 (c). Besides the structure in visual data, in Fig. 1.1 (d), we can find the textual descriptions have an associated structure with the corresponding images.

In this thesis, we focus on how to leverage the various forms of structure described above at different levels for visual understanding, generation, and reasoning:

- **Visual Understanding**. At the object-level, we propose a cross-channel communication module to enable communication across different channels at the same layer of Convolutional Neural Networks (CNNs). In this way, we try to learn a complementary set of filters, each of which focus on a specific part of object. At the image level, we propose an effective approach for scene graph generation. Based on the observation that scene graph is usually sparser, we propose to learn to prune the dense graph into a sparse one and then perform message passing on top of it. Finally at the dataset

(a) Instance-level Human Parsing in [1]

(b) COCO-stuff in [2]



(c) Visual Genome [3]

(d) Flickr30k [4]

Figure 1.1: Exemplar data in different datasets.

level, we propose a deep clustering method to simultaneously cluster the images and learn representations from them. Our method explicitly builds a k-nearest-neighbor graph for the image set and then evolves the graph gradually by merging similar images together.

- **Visual Generation**. As mentioned above, at the image level, an image usually consists of multiple objects. Most image generation methods generate the holistic images in one-shot. However, we argue that it would be beneficial to leverage an image structure prior in the image for generation. We propose a novel image generation model which generates the image in a compositional manner. Specifically, we first generate the image background and then the foreground objects one by one. Finally, we compose the image with all elements in a reasonable spatial configuration. This way we can avoid blending artifacts between the background and foreground.

- **Reasoning**. We further exploit the structured visual representation of images to bridge visual perception and reasoning to solve vision and language tasks. Specifically, instead of representing an image as a single feature vector, we propose to extract the image-level structure including objects, attributes and relationships from

a single image and then use them as the basis for down-stream vision and language tasks. We demonstrate the effectiveness through image captioning and visual question generation. By performing reasoning on the structured image representations, our models achieve better performance and generalization ability.

Below we review previous works related to our thesis and discuss how we exploit the structure in visual and textual data for visual understanding, generation and reasoning separately.

1.1 Visual Understanding

As discussed above, structures exist at different level. Below we will discuss how we can leverage such priors at different levels for visual understanding.

Object-level structure. The object-level structure has been exploited for decades even prior to the deep learning era. Particularly, Felzenszwalb et al [5, 6] proposed to train a multi-scale deformable part model for object detection. This method explicitly localize different object parts and then localize the whole object through an energy based optimization. Though it can achieve impressive results at that time, the features are hand-crafted (*e.g.*, HoG [7], SIFT [8], etc), and thus have limited representation power for complex object appearances. Recently, deep learning resolves this drawback. Using a simple CNN architecture, we can classify an image [9] or detect the objects [10] in an image without any extra efforts to design the features. It is believed that the convolutional layers can automatically learn hierarchical representations with a gradient-based optimization. Indeed, in [11, 12], we can see the convolutional layers can learn to localize the meaningful parts for image classification, object detection and even visual question answering. However, it is arguable that the redundancies still exist across different channels in a single convolutional layer, since a convolutional layer itself have no such explicit constraint. Motivated by this, our work is aimed at learning more diverse and complementary filters by building a cross-channel communication mechanism in a single convolution layer. We assume each

channel implicitly encode a specific part of the object, and the cross-channel communication calibrates the channels so that different channels learn to focus on different parts of object towards the final classification goal. We find such communication is effective to obtain more diverse representations and improve the image classification performance.

Image-level structure. A recent work [13] has proposed to represent images as graphs containing objects, their attributes, and the relationships between them. These *scene graphs* form an interpretable structured representation of the image that can support higher-level visual intelligence tasks such as captioning [14, 15], visual question answering [16, 17, 14, 18, 19, 20]. While scene graph representations hold tremendous promise, extracting scene graphs from images – efficiently and accurately – is challenging. The natural approach of considering every pair of nodes (objects) as a potential edge (relationship) – essentially reasoning over fully-connected graphs – is often effective in modeling contextual relationships but scales poorly (quadratically) with the number of objects, quickly becoming impractical. The naive fix of randomly sub-sampling edges to be considered is more efficient but not as effective since the distribution of interactions between objects is far from random. The work in this thesis is motivated by this natural property in scene graphs. Instead of predicting the relationships for all pairs of objects or ruling out most of the edges heuristically, a more intelligent way is learning to prune out the unlikely edges in the scene graphs before performing the scene graph labeling. This way the scene graphs become a sparse one and message passing upon it becomes more efficient and precise.

Dataset-level structure. We exploit the dataset-level strucutre for deep clustering. As the name indicates, deep clustering is a technique which combines deep representation learning and clustering on a set of unlabeled images. This final goal is to derive cluster labels $\mathbf{y}$ and representations $\mathcal{X}$. Thus far, we have seen a number of work in this direction [21, 22, 23, 24]. Among these works, ours is one of the earliest which attempted to combine clustering and representation learning seamlessly. Our basic argument is that a good representation is beneficial to clustering and vice versa. However, at the beginning, we by

no means can obtain either a good representation or initial clusters. Hence, we resort to agglomerative clustering [25] which regards each image as a unique cluster at the beginning. To determine how to merge the clusters, an affinity graph is built among different clusters in which two clusters are connected to each other if they are similar in the representation space, and otherwise disconnected. During the learning process, we gradually evolve the affinity graph for the image set by merging closest clusters at current hierarchy which in turn will be used to update the representations. This learning process proceeds alternatively until we reach the targeted cluster number. The results demonstrate the effectiveness of our proposed method on a variety of image datasets.

1.2 Visual Generation

Visual generation, as the dual problem of visual understanding, is aimed at synthesizing images or videos. Since the invention of generative adversarial networks (GAN) [26] and variational autoencoders (VAE) [27], visual generation has rapidly become a popular research area in computer vision community. In this line of work, DCGAN [28] proposes a canonical framework for image generation, which consists of a sequence of deconvolution layers as generator and convolution layers as the discriminator. Starting from simple digits or faces [29, 30] to recent photo-realistic large-scale images [31, 32], we have witnessed a significant development in this area. While those models achieve impressive results on some constrained datasets (*e.g.*, aligned faces, single-object images), it is still not satisfactory when the model is asked to generate images with multiple objects. As we discussed above, images usually have some structures, *i.e.*, an image usually contains background and multiple foreground objects, and these objects tend to have some interactions with each other. Motivated by this observation, this thesis presents a novel image generation method by explicitly considering the image-level structure, *i.e.*, background, foregrounds and their spatial relationships. It turns out that adding this structure priors improves the quality of generated images.

1.3 Reasoning

Reasoning on top of visual and textual data is one of the fundamental abilities in human intelligence [33]. It requires the agent to understand not only the visual and textual inputs, but also the underlying reasoning procedures. In this thesis, we particularly explored how we can exploit the symbolic structured representation at image-level for image captioning and visual question generation. By symbolic here, we mean the representation is either represented verbally, or a probability distribution over the vocabulary. In both image captioning and visual question generation, the models need to reason on top of the symbolic representations of images to predict which object to mention at next so that the overall captions are understandable, an the questions are informative.

Image captioning. Image captioning is aimed at generating a globally plausible description for a given image in terms of both groundingness and naturality. The groundingness here measures how much the generated textual descriptions align with the image content, while the naturality measures how much the descriptions align with human descriptions. Recently, most of state-of-the-art image captioning models use an encoder-decoder paradigm equipped with attention module [34, 35, 36]. Though these models can generate natural deceptions, it is well-understood that these models still lack visual grounding. One of the reasons is that the visual branch in these models still uses a holistic feature map without an explicit notion of structure in the image. Though the recent works such as [37] proposed to extract objects from the image, they used feature-level representations as the inputs to a recurrent neural network. In this thesis, however, we proposed to use symbolic representations instead to represent the objects in images. Note that this symbolic methods are not new in image captioning. In [38, 39, 40, 41], the models relied heavily on outputs of object detectors and attribute classifiers to describe images. In contrast to recent approaches, the language is unnatural but the caption is more grounded in what the model sees in the image. To achieve both groundingness and naturality, we propose a hybrid

method which takes the symbolic outputs of object detector and send it to a neural captioning module. It turns out this new method achieves a good trade-off between groundingness and naturality, and generalize well to compositionally novel scenarios.

Visual question generation. In this thesis, we consider an open-world scenario where a visual system is put in an environment with some novel visual concepts. In this case, one way for the agent to acquire information about the new concepts is by asking questions. There are a number of works in this area [42, 43, 44, 45, 46]. Tellex *et al*. [42] present a graphical model to address linguistic co-references. However, the question generation policy and vision system are not designed to learn. Lütkebohle *et al*. [43] propose to use language-interaction to solve ambiguities in the object references and for grasping commands. Recently, Thomason *et al*. [46] proposed to learn a dialog generation policy for natural language grounding. Instead of resolving the co-reference between visual data and text, [45] and [44] propose to generate questions for life-long continuous learning for new object concepts, which are close to our task. However, both of them simplify the testing scenario by assuming only one target object is presented to the agent or the agent can easily refer to the target object without ambiguity. In realistic environments however, there are usually multiple objects scattered in a single image. Our goal is for agents to learn a question generation policy that can ask *unambiguous* and *informative* questions to an oracle to learn a visual recognition system. This process involves an explicit reasoning on which entity should be asked next based on current state. Moreover, the question generation policy should be disentangled from the visual recognition system and specifics of the scenes so that it can be applied to other scenarios in the future. To achieve this, we extract a symbolic scene graph from an image and then generate the questions based on the scene graph. We demonstrate that the proposed model can learn to ask informative and unambiguous questions which helps the visual system to acquire more information. Moreover, the question generation policy can be easily generalized to novel scenarios.

1.4 Thesis Statement

Exploiting structure in visual and textual data enhances visual understanding, generation and reasoning. Specifically, we show that

- integrating the structure at object-level, image-level and dataset-level in visual data improves visual understanding (e.g., image classification, scene graph generation);
- exploiting the image-level structure to generate images in a layer-wise manner improves the visual quality of generated images;
- using symbolic structured representations of images not only improves the model performance but also increases the generalization ability in higher level reasoning tasks (e.g., vision and language tasks).

1.5 Thesis Overview

Part I introduces how we leverage object-level, image-level and dataset-level structures in the visual data for image classification, scene graph generation and representation learning, respectively; Part II explains how we leverage the structure existing in a single image for image generation; Part III illustrates how we reason on the symbolic structured representation of images for image captioning and visual question generation. Finally, in Chapter 8, we conclude and discuss some promising future directions.

Part I

Structured Visual Understanding

CHAPTER 2

LEVERAGE OBJECT-LEVEL STRUCTURE FOR IMAGE CLASSIFICATION

In this chapter, I will introduce how we leverage the object-level structure for improving the image classification. Specifically, we propose a new neural network module called cross-channel communication (C3) network, which enables the communication across different channels at the same convolutional layer in a convolutional neural network (CNN). In our experiments, we show that by stacking our C3 block on top of convolutional layers, the image classification performance is improved for various CNN architectures. We apply C3 block for other vision tasks such as object detection and image semantic segmentation and show similar trend of improvement. Furthermore, we perform ablation studies on the proposed C3 block and find it can learn more diverse representations and reduce the required network depth. Finally, the visualization results show that our C3 block can prompt the convolutional filters to pay attention to different object regions and learn a more complementary representations.

2.1 Introduction

The standard deep networks pass feature responses from lower-level layers to higher-level layers in a hierarchical fashion. With improved computational powers and novel network designs, stacking more layers has become a common and effective practice – the number of layers can be significantly large [9] and the connections between layers can be significantly dense [47]. Studies [48, 49] show that learned filters in the first few layers typically capture low-level texture in images, while the last few layers encode higher-level semantics. This structure is typically very effective for solving computer vision tasks, such as image classification [50], object detection [51, 9], semantic segmentation [52, 53] and video classification [54, 55, 56, 57].

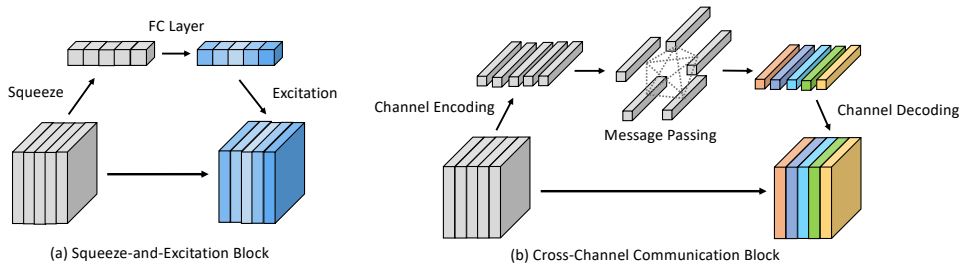


Figure 2.1: (a) network with Squeeze-and-Excitation (SE) block [58]; (b) our proposed cross-channel communication (C3) block. Without the squeeze operation as in SE block, our C3 block enables a comprehensive communication across different channels.

Consider a conventional convolutional neural network, filters at each layer typically respond to the input response independently. As a result, redundant information could be accumulated across different channels in the same convolutional layer. As a pioneer, Squeeze-and-Excitation (SE) block [58] (Figure.2.1 (a)) is an architecture to explicitly model the channel-wise interaction, and demonstrates superiority on various vision tasks. However, SE block average each channel into a single scalar and modulate the responses through a simple multiplication. Therefore, it has a relatively limited bandwidth for the channel-wise communication.

In this paper, we introduce Cross-channel Communication (C3) block, a simple yet effective operator that encourages information exchange across different channels at the same convolutional layer. In the C3 block, we first pass the response from each channel to a feature encoder, and then use a message passer to pass the information of one channel to all other channels. Then, we use a feature decoder to decode the message for each channel. The updated feature responses will then be passed to future layers and help perform downstream tasks. We provide a high-level overview of the C3 block in Figure.2.1 (b).

The proposed module allows channels in each layer to communicate with each other before passing information to the next layer. Different from related network designs [58, 59, 60] where channels within each layer have limited interactions, our module enables channels to have a comprehensive interaction through a fully-connected graph structure. In C3 block, we retain the response map (no squeeze used) so that each channel has information

of where *and* how the other channels respond to specific patterns in the image (e.g., different body parts of a person), and then introduce the feature encoder and decoder to enable thorough information exchange across channels, to learn complementary representations.

Experimental results demonstrate that the learned features are more effective for downstream computer vision tasks such as image classification, semantic segmentation and object detection. To further validate our claim, we conduct experiments to analyze the behavior of C3 block. We find that 1) the correlations among channels in each layer are smaller than the baseline model, suggesting that filters in each layer can learn a more diverse set of representations; 2) when applying it to a shallower network, we can still achieve similar performance to deeper networks without C3 block, indicating that the learned features under C3 blocks are more representative.

2.2 Cross-channel Communication Unit

2.2.1 Formulation

In this section, we formally introduce the formulation of channel-wise interaction. Using 2D CNNs as an example, we illustrate this network module in Figure 2.2.

Let us consider a L-layer neural network architecture, where each layer has n_l filters. In the l-th layer, the feature responses are denoted by $\boldsymbol{X}_l = \{\boldsymbol{x}_l^1, ..., \boldsymbol{x}_l^{n_l}\}$. In CNNs, the response of each channel is a $H_l \times W_l$ feature map where H_l and W_l denotes spatial resolutions. We can formulate the updated response after performing the channel-wise interaction by

$$\hat{\boldsymbol{x}}_l^i = \boldsymbol{x}_l^i + f_l^i(\boldsymbol{x}_l^1, ..., \boldsymbol{x}_l^{n_l}) \tag{2.1}$$

where f_l^i is a function that takes all the feature responses at all channels, and updates the encoded features of a particular channel. We define *cross-channel communication* as the exchanged information across all channels, and the formulation of such information is modeled in f_l^i. In Squeeze-and-Excitation block [58], f_l represents a squeeze and excitation

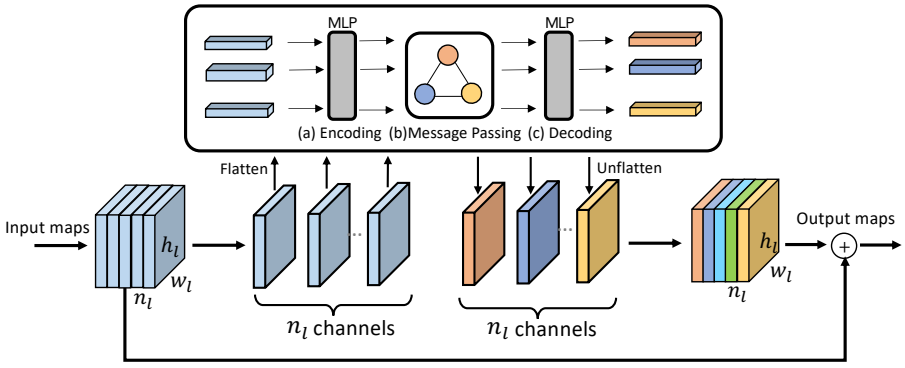


Figure 2.2: An overview of the Cross-channel Communication (C3) block. The feature responses in channels are passed to an encoder, and then the information is exchanged with other channels using a message passing mechanism. Finally, the features are decoded and added back to the input responses for the recalibration.

operation [1]. In comparison, our proposed network structure enables a more comprehensive communication through a graph neural network by treating each channel as a node in the graph. We will discuss the details of the model in the following section.

2.2.2 Architecture

The proposed cross-channel communication network consists of three parts that are used for feature encoding, message passing and feature decoding, respectively.

Feature encoding. This module is used to extract the global information from each channel response map. Specifically, given the response map $\boldsymbol{x}_l^i$, we first flatten it to be an one-dimensional feature vector, and then pass it to two fully-connected layers:

$$\boldsymbol{y}_l^i = f_{enc}^{in}(\boldsymbol{x}_l^i), \quad \boldsymbol{z}_l^i = f_{enc}^{out}(\sigma(\boldsymbol{y}_l^i)) \tag{2.2}$$

In our model, f_{enc}^{in} and f_{enc}^{out} are two linear functions and σ is a Rectified Linear Unit (ReLU).

In the feature encoding module, we add a bottleneck after f_{enc}^{in} to reduce the feature

[1]The original SE block does not have a residual addition, but we can reformulate it into a residual version by subtracting the scalar multiplier for each channel by 1

dimension by a factor of $\alpha > 1$. This module compresses the features to reduce the computational cost. We set $\alpha = 8$ in our experiments.

Message Passing. This module enables channels to interact with each other, and update their feature responses. For modeling such interactions, graph convolutional network [61] is a typical choice. Recently, graph attention networks [62, 63] have also been introduced, aiming to build a soft attention mechanism on top of GCNs. We use a similar formulation to model the cross-channel communication. Specifically, we construct a complete undirected graph, where $\boldsymbol{Z} = \{\boldsymbol{z}_l^i\}$ are nodes. We denote the edge strength between the two nodes to be $s_{ij} = f_{\text{att}}(\boldsymbol{z}_l^i, \boldsymbol{z}_l^j)$. Recent works use various methods to learn f_{att} [56, 62], while we use a simple yet effective way to compute the edge strength:

$$\bar{z}_l^i = \sum_{k=1}^{h_l w_l} \boldsymbol{z}_l^i[k]/(h_l w_l), \quad s_{ij} = -(\bar{z}_l^i - \bar{z}_l^j)^2 \tag{2.3}$$

where $\boldsymbol{z}_l^i[k]$ is the k-th element of the flattened vector $\boldsymbol{z}_l^i$. In above, we use the average output from the feature encoder to increase the robustness in message passing period. We then compute the negative square distance to enable the channels with similar properties to have more communication, through which we want to group the similar channels and then make them diverse and complementary. We then feed them to a softmax layer to get the normalized attention scores a_{ij}, and then compute the output $\hat{\boldsymbol{Z}} = \{\hat{\boldsymbol{z}}_l^i\}$ as follows:

$$\hat{\boldsymbol{z}}_l^i = \sum_{j=1}^{n_l} a_{ji} \boldsymbol{z}_l^j \tag{2.4}$$

Feature Decoding. After acquiring the updated channel-wise outputs $\hat{\boldsymbol{Z}}$, the decoding module takes this information that contains the corrected beliefs of all channels, and reshape it to the same size of the input volume so that it can be passed to future layers. We will add back this information to $\boldsymbol{X}_l$ as shown in Eq. (2.1).

The above mechanism ensures that channels at the same layer can communicate with

each other comprehensively before passing information to future layers. The encoding layer captures the high-level information of each channel, the message massing module enables channels to interact with each other, and the decoding module collects the information and passes it to subsequent layers.

Model and Computational Complexity. Our module has a relatively low computational complexity. In each block, the computation only involves four FC layers. For the NC block at layer l, the additional parameters introduced is $N_l = \frac{4}{\alpha}\sum_{l=1}^{L}(H_l W_l)^2$, where α is the reduction ratio in our bottleneck layer as mentioned above. As a result, our module is independent of the number of channels, and thus introduces reasonable number of parameters for both small and large networks. In practice, we find it is not necessary to add cross-channel communication at all convolutional layers. Hence, the complexity is further limited by adding our C3 block to only a few separate layers.

2.2.3 Analyzing the Channel Responses

To understand how the communication affect the channel response maps, we compute the correlations among all the response maps within each layer, and compare the behavior with/without using NC block. Specifically, at each spatial location $[m, n]$ in the feature map, we compute the correlations:

$$c_l^{ij} = \frac{\sum_{m=1}^{H_l}\sum_{n=1}^{W_l}(x_l^i[m,n] - \bar{x}_l^i)(x_l^j[m,n] - \bar{x}_l^j)}{\sigma_{x_l^i}\sigma_{x_l^j}} \tag{2.5}$$

where $\bar{x}_l^i$ and $\sigma_{x_l^i}$ are the mean and standard derivation of $\boldsymbol{x}_l^i$. We then take the absolute values of all the c_l^{ij}, and take the average. A larger value indicates that there is redundancy in the encoded features, while a smaller value means the learned features are more diverse.

2.3 Background

Our design of the cross-channel communication network shares some high-level similarities with some recently proposed network units. We highlight a few most related works, and discuss their the differences. Broadly, we categorize these networks into two categories: modeling feature map interactions spatially or among each channel.

Networks that Model Spatial Interactions. There are network structures that learn spatial transformation to change input feature maps [64, 65]. Besides learning to perform spatial transformation, Wang *et al*.proposed a non-local network (NLN) to describe the inter-dependency in long-range spatial-temporal locations [56]. We also use graph neural network to model context within a single layer. However, NLN still models interactions for features spatially and primarily works for video data. In contrast, we model features within each channel. $\boldsymbol{z}_l$ in Eq. (2.4) is updated using information from other channels, whereas each element of $\boldsymbol{z}_l$ is updated using its own elements in NLN.

Networks that Model Channel-Wise Interactions. The Squeeze-and-Excitation Network [58] falls into this category in that it uses a simple network to calibrate feature responses. Besides performing a channel-wise scaling, [59, 60] proposed channel-wise attention for image captioning or semantic segmentation. In our formulation, the interactions across channels are modeled through a more comprehensive yet efficient mechanism. Concretely, in Eq. (2.4), SE block can be viewed as $\hat{\boldsymbol{z}}_l = a_l \boldsymbol{z}_l$. Similarly, the layer Normalization [66] network is yet another layer-wise operation, but with even simpler operations. More generally, various normalization methods such as group normalization [67] can be also regarded as a special case of channel-wise communication. However, the interactions across different channels are less powerful in that only a mean feature map and the standard deviation are learned.

2.4 Experiments

To demonstrate the effectiveness of the proposed module, we conduct experiments by plugging it into various network architectures to enable cross-channel communication within a layer. We mainly use the residual network [9] and its variants for our experiments. For clarity, we denote the union of all residual blocks with the same feature resolution as a *residual layer*. We first evaluate our model on image classification tasks, then verify its generalization ability to other tasks including semantic segmentation and object detection. Finally, we analyze the model behavior through ablation studies and visualizations.

2.4.1 Quantitative Comparison

Image Classification. We conduct experiments on two popular benchmarks: 1) CIFAR-100 [68], which has 100 object classes, and 500 images each for training and 100 for testing. (2) ImageNet [69], which has 1000 classes and more than 1.28M images for training, and 50K for validation.

We use representative network structures including ResNet [9], and Wide-ResNet [70]. We also compare our proposed module with the Squeeze-and-Excitation (SE) block [58]. For a trade-off between model complexity and performance, we add our C3 block to a few separate layers. Specifically, for both ResNet and Wide-ResNet, we add one C3 block to each residual layer at the front. For a fair comparison, we use publicly available code and the same training protocols (including data loader, learning rate and schedule, optimizer, weight decay, training duration) for all models. Specifically, we use stochastic gradient descent (SGD) with an initial learning rate 0.1, momentum 0.99, and weight decay $1\mathrm{e}{-4}$ for both datasets. The learning rate is decayed by 10 after 100 and 140 epochs for CIFAR-100, and 30 and 60 for ImageNet. We report the average best accuracy of 5 runs.

Table 2.1 shows the classification errors on CIFAR-100 for different models and network architectures and the corresponding model size (in millions). Our proposed cross-

	ResNet-20			ResNet-56			ResNet-110			Wide-ResNet		
	Size	FLOPs	Acc.	Size	FLOPs	Acc.	Size	FLOPs	Acc.	Size	FLOPs	Acc.
Baseline	0.28	41.7	67.73	0.86	128.2	71.05	1.74	257.9	72.01	26.86	3.84G	77.96
Baseline + SE	0.28	41.8	68.57	0.87	128.5	72.00	1.76	258.5	72.47	27.26	3.84G	**78.57**
Baseline + C3	0.35	46.0	**69.34**	0.93	132.5	**72.27**	1.81	262.2	**73.36**	26.93	3.87G	78.34

Table 2.1: Classification accuracies (%) on CIFAR-100 [68] with different models.

	ResNet-18			ResNet-50			ResNet-101		
	size	top-1 err.	top-5 err.	size	top-1 err.	top-5 err.	size	top-1 err.	top-5 err.
Baseline	11.69	30.28	10.52	25.56	23.61	7.27	44.55	22.48	6.18
Baseline + SE	11.78	30.15	10.72	28.07	**22.51**	**6.43**	49.29	22.14	6.14
Baseline + C3	12.02	**29.30**	**10.48**	25.89	23.19	6.60	44.88	**21.93**	**6.02**

Table 2.2: Classification errors on ImageNet [69] with different models.

channel communication module consistently outperforms the baseline and SE block [58] on various ResNet architectures. On Wide-ResNet, our network outperforms baseline model and is on par with SE network while introducing much fewer parameters (0.07M versus 0.40M). Note that on all these network architectures, our module introduces the same number of parameters because it is independent of the model size.

In Table 2.2, we report the classification errors on ImageNet for different models along with the model size (in millions). We use standard ResNet-18, ResNet-50 and ResNet-101 as baselines for comparisons. We observe a consistent trend as in Table 2.1. Our cross-channel communication module outperforms baseline consistently, by introducing only 0.33M parameters. Meanwhile, our model achieves comparable performance to SE block, even though it introduces more parameters (over 3M parameters). In our experiments, we also observe that models with cross-channel communication modules consistently have lower errors in both training and validation over the whole training period.

Object Detection and Semantic Segmentation. We use Faster R-CNN [51] for object detection on the COCO dataset [71], and Deeplab-V2 [53] for semantic segmentation on the Pascal VOC dataset [72]. We refer to [73] for the implementation of Faster R-CNN. We add the C3 block in those network structures, and report scores in Table 2.3. Specifically, for semantic segmentation, similar to the image classification task, we append one C3 block

to each of the residual layers. For object detection, we only add one C3 block to the output of ROI pooling layer. We can see consistent improvements for two tasks. Recall that we only introduce a few additional parameters. Note that we train both models in a plug-and-play manner, which is different from the experimental settings in [74]. The goal of these experiments is to prove the generalization of our module in various tasks.

Segmentation	Mean IOU	Mean Acc.
Deeplabv2 [53]	75.2	85.3
Deeplabv2 + SE	**75.6**	85.6
Deeplabv2 + C3	**75.7**	**86.0**

Detection	Pascal VOC	COCO
Faster R-CNN [51]	74.6	33.9
Faster R-CNN + SE	74.8	34.3
Faster R-CNN + C3	**75.6**	**34.8**

Table 2.3: Performance on semantic segmentation on PASCAL-VOC-2012 (left) and object detection on PASCAL-VOC-2007 and COCO (right) with/without cross-channel communication). Bold indicates best results. For detection, mAP@(IOU=0.5) is reported for PASCAL-VOC-2007 and mAP@(IOU=0.5:0.95) is reported for COCO.

2.4.2 Analyzing the Communication Block

In this section, we systematically investigate the behavior of the C3 block from different aspects. Specifically, we will answer the following questions.

Can we reduce the depth of the network when using C3 block? Since channels at each layer can communicate and interact through our C3 block, one assumption is that a very deep network is not necessary to propagate information across channels. Therefore, we conduct experiments by adding only a few C3 blocks while reducing the depth of the neural network. We perform this ablated experiment on CIFAR-100 for image classification using ResNet [9] architecture with different number of residual blocks at each residual layer. In Figure 2.3, we see that using the C3 block, a shallower ResNet-74 can perform on par with ResNet-110 without C3 block, though with fewer parameters. This suggests that our C3 module can help to reduce the depth of neural networks while retaining the performance. As a reference, we further report the detailed number of parameters in these network structures in Table 2.4. As we can see, the networks with C3 block achieve bet-

ter performance though having a similar amount of parameters to the baseline network. Through this ablation study, we demonstrate that the improvement of our model is not simply due to the increase in parameters.

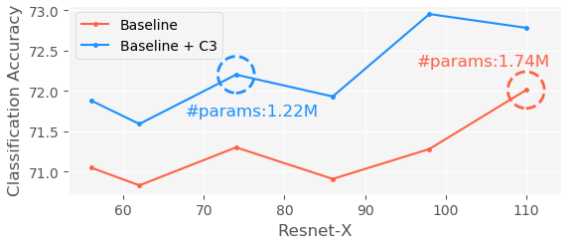


Figure 2.3: The classification accuracy for ResNet with different number of layers.

	W/O C3	W/ C3
ResNet-56	0.86	0.93
ResNet-62	0.96	1.03
ResNet-74	1.15	1.22
ResNet-86	1.35	1.41
ResNet-98	1.54	1.61
ResNet-110	1.74	1.80

Table 2.4: Model size for different ResNets.

Does C3 block reduces redundancy in features? To understand the outcome of cross-channel communication, we compute the correlation scores (as described in Sec 2.2.3) among all the channel-wise features for the models without and with our C3 blocks. We track the correlation of channel responses during the whole training stage. As indicated in Fig. 2.4, after features are passed to the proposed module (Baseline + C3 After), the correlations among channels is consistently smaller (Baseline + C3 Before). When comparing to the baseline, both values are significantly smaller. This suggest that channels are effectively communicating with each other so that the encoded features become less redundant, and hence achieve a better performance.

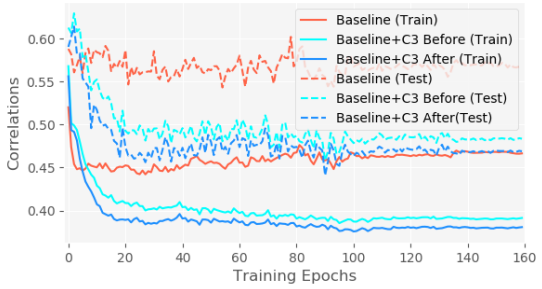


Figure 2.4: Correlations for models at different training stages.

E-D	M-P	ResNet-20	ResNet-56	ResNet-110
-	-	67.73	71.05	72.01
✓	-	68.70	71.95	72.65
-	✓	69.13	71.79	72.74
✓	✓	**69.34**	**72.27**	**73.36**

Table 2.5: Classification accuracy on CIFAR-100 for ablated C3 block. E-D is "Encoder and Decoder"; M-P is "Message Passing".

Is our model design helpful? We investigate the extent to which the feature encoding/decoding and message passing contribute to the performance improvement. Specifi-

cally, we remove either the feature encoder/decoder or message passing from our C3 block, and perform image classification on CIFAR-100. As we can see in Table 2.5, both feature encoding/decoding and message passing improve the performance over the baseline network. These results demonstrate that both modules are necessary. The message passing helps channels to exchange information across channels so that each channel has a global information about the input. Without communication, the feature encoding and decoding help each channel to capture its own global structural information that can not be captured by a single or few convolution layers.

Where should we add the C3 block in the network? In CNNs, the lower-level layers typically encode low-level image features while higher-level layers contain semantic information [48]. In this experiment, we investigate the effect of adding the C3 block at different residual layers of ResNet. As indicated in Table 2.6, adding C3 blocks at the second or third residual layer is typically more effective. Note that due to larger feature size, the C3 block at first residual layer has more parameters than those in second and third residual layer. This further supports our previous claim that the improvement is not entirely due to the increase of model size. Instead, the C3 block at the second and third residual layer indeed learns more helpful information to help the task. An explanation is that filters at higher layer typically encode high-level semantic information [48], it's more likely get diverse and informative responses through communication. This indicates that we can further reduce the model size with few sacrifice of performance, by merely adding our C3 block in the last few layers.

layer 1	layer 2	layer 3	ResNet-20	ResNet-56	ResNet-110
✓	-	-	68.67	71.16	72.28
-	✓	-	**69.53**	71.88	72.78
-	-	✓	69.12	71.53	72.01
-	✓	✓	69.19	72.03	72.95
✓	✓	✓	69.34	**72.27**	**73.36**

Table 2.6: Classification accuracy for models with C3 block at different residual layers.

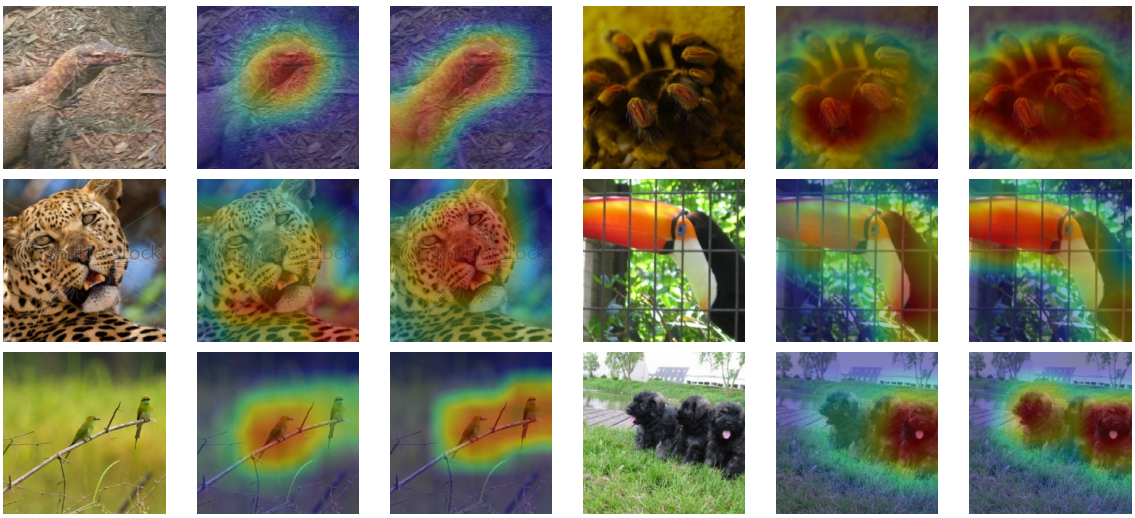


Figure 2.5: Class Activation Maps (CAM) at the last layer for ResNet-101 (2nd and 5th column) and ResNet-101 with C3 block (3rd and 6th column).

2.4.3 Visualizations

We have shown in Fig. 2.4 that the proposed C3 block can help channels to learn more diverse representations. To further investigate what the C3 block has learned, we employ an off-the-shelf tool to visualize the class activation map (CAM) [11]. We use Resnet-101 and Resnet-101+C3 trained on ImageNet for comparison. As shown in first and second row of Fig. 2.5, the heat maps extracted from CAM for our model have more coverage on the object region, and less coverage on the background region. In the bottom row, for images which contain multiple objects, the filters learned from our model can localize all the objects, while the original model usually localize at most salient object region in the image. Furthermore, we show the top six mostly intense class activation maps from the last layer. This way we can directly check what each channel excites about and where they are. As shown in Fig. 2.6, the top activation maps from baseline model have more overlaps than the activation maps from the model with C3 block. These visualizations further demonstrate that C3 block can help the channels to learn more comprehensive and complementary filters.

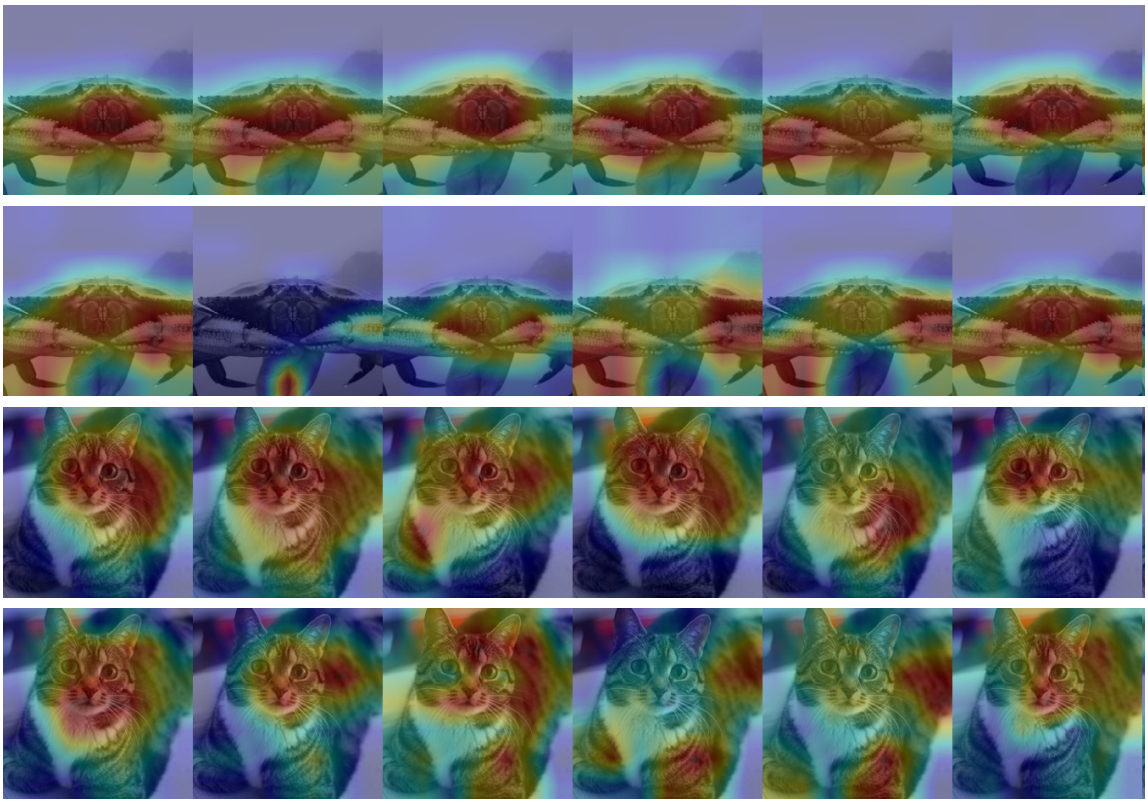


Figure 2.6: We visualize the top-6 mostly activated channels at the last layer. Odd rows are the class activation map from baseline ResNet-101; Even rows are from our model.

2.5 Discussion

In this paper, we introduced a novel network unit, called Cross-channel Communication (C3) block. Unlike standard hierarchical deep network architectures, we allow channels in each layer to communicate with each other. Through communication, channels at the same layer can capture global information and calibrate with other channels. To encourage the communication, we use a simple yet effective graph neural network that consists of a feature encoder, a message passing step, and a feature decoder. Our experimental results and ablation studies demonstrate that the C3 blocks can be added to modern network structures to advance the performance for a variety of computer vision tasks, with a small burden of model parameters.

CHAPTER 3

LEVERAGE IMAGE-LEVEL STRUCTURE FOR SCENE GRAPH GENERATION

In this chapter, I will introduce how we leverage the image-level structure for scene graph generation. To generate a scene graph, we need to extract both the objects and relationships between objects from a single image. However, not all objects have the relationships with each other which means the scene graph is usual sparse. Based on this observation, we propose a relation proposal network (RePN) which learns to prune a densely-connected scene graph to a sparse one. Afterward, we propose an attentional GCN (aGCN) to perform message passing across different nodes in the scene graph. Experimental results show that the proposed RePN cannot only improve the scene graph generation performance but also the object detection performance. Stacking the aGCN on top of model can further improve the scene graph generation performance.

3.1 Introduction

Visual scene understanding has traditionally focused on identifying *objects in images* – learning to predict their presence (*i.e*. image classification [75, 50, 9]) and spatial extent (*i.e*. object detection [76, 77, 78] or segmentation [79]). These object-centric techniques have matured significantly in recent years, however, representing scenes as collections of objects fails to capture relationships which may be essential for scene understanding.

A recent work [13] has instead proposed representing visual scene as graph containing objects, their attributes, and the relationships between them. This *scene graph* forms an interpretable structured representation of the image that can support higher-level visual intelligence tasks such as captioning [14, 15], visual question answering [16, 17, 14, 18, 19, 20], and image-grounded dialog [80]. While scene graph representations hold tremendous promise, extracting scene graphs from images – efficiently and accurately – is challeng-

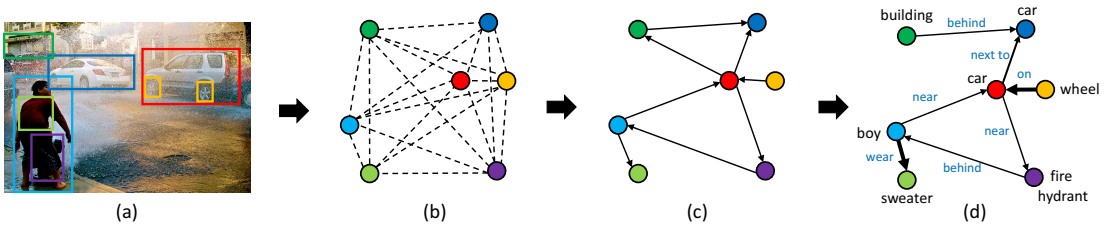


Figure 3.1: Given an image (a), our proposed approach first extracts a set of objects visible in the scene and considers possible relationships between all nodes (b). Then it prunes unlikely relationships using a learned measure of 'relatedness', producing a sparser candidate graph structure (c). Finally, an attentional graph convolution network is applied to integrate global context and update object node and relationship edge labels.

ing. The natural approach of considering every pair of nodes (objects) as a potential edge (relationship) – essentially reasoning over fully-connected graphs – is often effective in modeling contextual relationships but scales poorly (quadratically) with the number of objects, quickly becoming impractical. The naive fix of randomly sub-sampling edges is more efficient but not as effective since the distribution of interactions between objects is far from random – take Fig. 3.1(a) as an example, it is much more likely for a 'car' and 'wheel' to have a relationship than a 'wheel' and 'building'. Furthermore, the types of relationships that typically occur between objects are also highly dependent on objects.

Graph R-CNN. In this work, we propose a new framework, Graph R-CNN, for scene graph generation which effectively leverages regularities through two mechanisms to intelligently sparsify and reason over candidate scene graphs. Our model can be factorized into three logical stages: 1) object node extraction, 2) relationship edge pruning, and 3) graph context integration, which are depicted in Fig. 3.1. In the object node extraction stage, we utilize a standard object detection pipeline [51]. This results in a set of localized object regions as shown in Fig. 3.1b. We introduce two important novelties in the rest of the pipeline to incorporate the real-world regularities in object relationships discussed above. First, we introduce a relation proposal network (RePN) that learns to efficiently compute *relatedness scores* between object pairs which are used to intelligently prune unlikely scene graph

connections (as opposed to random pruning in prior work). A sparse post-pruning graph is shown in Fig. 3.1c. Second, given the resulting sparsely connected scene graph candidate, we apply an attentional graph convolution network (aGCN) to propagate higher-order context throughout the graph – updating each object and relationship representation based on its neighbors. In contrast to existing work, we predict per-node edge attentions, enabling our approach to learn to modulate information flow across unreliable or unlikely edges. We show refined graph labels and edge attentions (proportional to edge width) in Fig. 3.1d.

To validate our approach, we compare our performance with existing methods on the Visual Genome [81] dataset and find that our approach achieves an absolute gain of 5.0 on Recall@50 for scene graph generation [82]. We also perform extensive model ablations and quantify the impact of our modeling choices.

Evaluating Scene Graph Generation. Existing metrics for scene graph generation are based on recall of ⟨subject, predicate, object⟩ triplets (e.g. `SGGen` from [81]) or of objects and predicates given ground truth object localizations (e.g. `PredCls` and `PhrCls` from [81]). In order to expose a problem with these metrics, consider a method that mistakes the boy in Fig. 3.1a as a man but otherwise identifies that he is 1) standing behind a fire hydrant, 2) near a car, and 3) wearing a sweater. Under the triplet-based metrics, this minor error (boy vs man) would be heavily penalized despite most of the boy's relationships being correctly identified. Metrics that provide ground-truth regions side-step this problem by focusing strictly on relationship prediction but cannot accurately reflect the test-time performance of the entire scene graph generation system.

To address this mismatch, we introduce a novel evaluation metric (`SGGen+`) that more holistically evaluates the performance of scene graph generation with respect to objects, attributes (if any), and relationships. Our proposed metric `SGGen+` computes the total recall for singleton entities (objects and predicates), pair entries ⟨object, attribute⟩ (if any), and triplet entities ⟨subject, predicate, object⟩. We report results on existing methods under

this new metric and find our approach also outperforms the state-of-the-art significantly. More importantly, this new metric provides a more robust and holistic measure of similarity between generated and ground-truth scene graphs.

Summary of Contributions. Concretely, this work addresses the scene graph generation problem by introducing a novel model (Graph R-CNN), which can leverage object-relationship regularities, and proposes a more holistic evaluation metric (`SGGen+`) for scene graph generation. We benchmark our model against existing approaches on standard metrics and this new measure – outperforming existing approaches.

3.2 Background

Contextual Reasoning and Scene Graphs. The idea of using context to improve scene understanding has a long history in computer vision [83, 84, 85, 86]. More recently, inspired by representations studied by the graphics community, Johnson *et al*.[13] introduced the problem of extracting scene graphs from images, which generalizes the task of object detection [76, 10, 51, 77, 78] to also detecting relationships and attributes of objects.

Scene Graph Generation. A number of approaches have been proposed for the detection of both objects and their relationships [87, 88, 89, 90, 91, 82, 92, 93, 94, 95, 96, 97]. Though most of these works point out that reasoning over a quadratic number of relationships in the scene graph is intractable, each resorted to heuristic methods like random sampling to address this problem. Our work is the first to introduce a trainable relationship proposal network (RePN) that learns to prune unlikely relationship edges from the graph without sacrificing efficacy. RePN provides high-quality relationship candidates, which we find improves overall scene graph generation performance.

Most scene graph generation methods also include some mechanisms for context propagation and reasoning over a candidate scene graph in order to refine the final labeling. In [82], Xu *et al*.decomposed the problem into two sub-graphs – one for objects and one for

relationships – and performed message passing. Similarly, in [92], the authors propose two message-passing strategies (parallel and sequential) for propagating information between objects and relationships. Dai *et al*.[94] address model the scene graph generation process as inference on a conditional random field (CRF). Newell *et al*. [96] proposed to directly generate scene graphs from image pixels without the use of object detector based on associative graph embeddings. In our work, we develop a novel attentional graph convolutional network (aGCN) to update node and relationship representations by propagating context between nodes in candidate scene graphs – operating both on visual and semantic features. While similar in function to the message-passing based approach above, aGCN is highly efficient and can learn to place attention on reliable edges and dampen the influence of unlikely ones.

A number of previous approaches have noted the strong regularities in scene graph generation which motivate our approach. In [87], Lu *et al*. integrates semantic priors from language to improve the detection of meaningful relationships between objects. Likewise, Li *et al*. [95] demonstrated that region captions can also provide useful context for scene graph generation. Most related to our motivation, Zeller *et al*.[97] formalize the notion of motifs (*i.e*., regularly occurring graph structures) and examine their prevalence in the Visual Genome dataset [81]. The authors also propose a surprisingly strong baseline which directly uses frequency priors to predict relationships – explicitly integrating regularities in the graph structure.

Relationship Proposals. Our Relationship Proposal Network (RePN) is inspired and relates strongly to the region proposal network (RPN) of faster R-CNN [51] used in object detection. Our RePN is also similar in spirit to the recently-proposed relationship proposal network (Rel-PN) [98]. There are a number of subtle differences between these approaches. The Rel-PN model independently predicts proposals for subject, objects and predicates, and then re-scores all valid triples, while our RePN generates relations conditioned on ob-

jects, allowing it to learn object-pair relationship biases. Moreover, their approach is class agnostic and has not been used for scene graph generation.

Graph Convolutional Networks (GCNs) . GCNs were first proposed in [61] in the context of semi-supervised learning. GCNs decompose complicated computation over graph data into a series of localized operations (typically only involving neighboring nodes) for each node at each time step. The structure and edge strengths are typically fixed prior to the computation. For completeness, we note that an upcoming publication [62] has concurrently and independently developed a similar GCN attention mechanism (as aGCN) and shown its effectiveness in other (non-computer vision) contexts.

3.3 Approach

In this work, we model scene graphs as graphs consisting of image regions, relationships, and their labellings. More formally, let I denote an image, V be a set of nodes corresponding to localized object regions in I, $E \in \binom{V}{2}$ denote the relationships (or edges) between objects, and O and R denote object and relationship labels respectively. Thus, the goal is to build a model for $P(S = (V, E, O, R)|I)$. In this work, we factorize the scene graph generation process into three parts:

$$P(\boldsymbol{\mathcal{S}}|\boldsymbol{I}) = \overbrace{P(\boldsymbol{V}|\boldsymbol{I})}^{\text{Object Region Proposal}} \underbrace{P(\boldsymbol{E}|\boldsymbol{V}, \boldsymbol{I})}_{\text{Relationship Proposal}} \overbrace{P(\boldsymbol{R}, \boldsymbol{O}|\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{I})}^{\text{Graph Labeling}} \tag{3.1}$$

which separates graph construction (nodes and edges) from graph labeling. The intuition behind this factorization is straightforward. First, the object region proposal $P(\boldsymbol{V}|\boldsymbol{I})$ is typically modeled using an off-the-shelf object detection system such as [51] to produce candidate regions. Notably, existing methods typically model the second relationship proposal term $P(\boldsymbol{E}|\boldsymbol{V}, \boldsymbol{I})$ as a uniform random sampling of potential edges between vertices $\boldsymbol{V}$. In

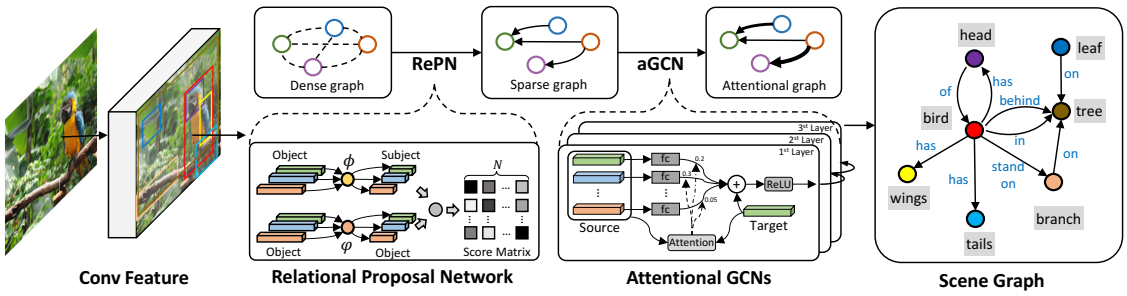


Figure 3.2: The pipeline of our proposed Graph R-CNN framework. Given an image, our model first uses RPN to propose object regions, and then prunes the connections between object regions through our relation proposal network (RePN). Attentional GCN is then applied to integrate contextual information from neighboring nodes in the graph. Finally, the scene graph is obtained on the right side.

contrast, we propose a relationship proposal network (RePN) to directly model $P(\boldsymbol{E}|\boldsymbol{V}, \boldsymbol{I})$ – making our approach the first that allows for learning the entire generation process end-to-end. Finally, the graph labeling process $P(\boldsymbol{R}, \boldsymbol{O}|\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{I})$ is typically treated as an iterative refinement process [92, 82, 94]. A brief pipeline is shown in Fig. 3.2.

In the following, we discuss the components of our proposed Graph R-CNN model corresponding to each of the terms in Eq. 3.1. First, we discuss our use of Faster R-CNN [51] for node generation in Section 3.3.1. Then in Section 3.3.2 we introduce our novel relation proposal network architecture to intelligently generate edges. Finally, in Section 3.3.3 we present our graph convolutional network [61] with learned attention to adaptively integrate global context for graph labeling.

3.3.1 Object Proposals

In our approach, we use the Faster R-CNN [51] framework to extract a set of n object proposals from an input image. Each object proposal i is associated with a spatial region $r_i^o = [x_i, y_i, w_i, h_i]$, a pooled feature vector x_i^o, and an initial estimated label distribution p_i^o over classes $C=\{1, \ldots, k\}$. We denote the collection of these vectors for all n proposals as the matrices $R^o \in \mathbb{R}^{n\times 4}$, $X^o \in \mathbb{R}^{n\times d}$, and $P^o \in \mathbb{R}^{n\times|C|}$ respectively.

3.3.2 Relation Proposal Network

Given the n proposed object nodes from the previous step, there are $O(n^2)$ possible connections between them; however, as previously discussed, most object pairs are unlikely to have relationships due to regularities in real-world object interactions. To model these regularities, we introduce a relation proposal network (RePN) which learns to efficiently estimate the *relatedness* of an object pair. By pruning edges corresponding to unlikely relations, the RePN can efficiently sparsify the candidate scene graph – retaining likely edges and suppressing noise introduced from unlikely ones.

In this paper, we exploit the estimated class distributions (P^o) to infer relatedness – essentially learning soft class-relationships priors. This choice aligns well with our intuition that certain classes are relatively unlikely to interact compared with some other classes. Concretely, given initial object classification distributions P^o, we score all $n * (n-1)$ directional pairs $\{\boldsymbol{p}_i^o, \boldsymbol{p}_j^o | i \neq j\}$, computing the relatedness as $s_{ij} = f(\boldsymbol{p}_i^o, \boldsymbol{p}_j^o)$ where $f(\cdot, \cdot)$ is a learned relatedness function. One straightforward implementation of $f(\cdot, \cdot)$ could be passing the concatenation $[\boldsymbol{p}_i^o, \boldsymbol{p}_j^o]$ as input to a multi-layer perceptron which outputs the score. However, this approach would consume a great deal of memory and computation given the quadratic number of object pairs. To avoid this, we instead consider an asymmetric kernel function:

$$f(\boldsymbol{p}_i^o, \boldsymbol{p}_j^o) = \langle \Phi(\boldsymbol{p}_i^o), \Psi(\boldsymbol{p}_j^o) \rangle, i \neq j \tag{3.2}$$

where $\Phi(\cdot)$ and $\Psi(\cdot)$ are projection functions for subjects and objects in the relationships respectively[1]. This decomposition allows the score matrix $S = \{s_{ij}\}^{n \times n}$ to be computed *with only two projection processes for X^o followed by a matrix multiplication*. We use two multi-layer perceptrons (MLPs) with identical architecture (but different parameters) for $\Phi(\cdot)$ and $\Psi(\cdot)$. We also apply a sigmoid function element-wise to S such that all relatedness scores range from 0 to 1.

[1] We distinguish between the first and last object in a relationship as subject and object respectively, that is, $\langle$`subject`, `relationship`, `object`$\rangle$.

After obtaining the score matrix for all object pairs, we sort the the scores in descending order and choose top K pairs. We then apply non-maximal suppression (NMS) to filter out object pairs that have significant overlap with others. Each relationship has a pair of bounding boxes, and the combination order matters. We compute the overlap between two object pairs $\{u, v\}$ and $\{p, q\}$ as:

$$IoU(\{u, v\}, \{p, q\}) = \frac{I(r_u^o, r_p^o) + I(r_v^o, r_q^o)}{U(r_u^o, r_p^o) + U(r_v^o, r_q^o)} \tag{3.3}$$

where operator I computes the intersection area between two boxes and U the union area. The remaining m object pairs are considered as candidates having meaningful relationships $\boldsymbol{E}$. With $\boldsymbol{E}$, we obtain a graph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$, which is much sparser than the original fully connected graph. Along with the edges proposed for the graph, we get the visual representations $X^r = \{\boldsymbol{x}_1^r, ..., \boldsymbol{x}_m^r\}$ for all m relationships by extracting features from the union box of each object pair.

3.3.3 Attentional GCN

To integrate contextual information informed by the graph structure, we propose an attentional graph convolutional network (aGCN). Before we describe our proposed aGCN, let us briefly recap a 'vanilla' GCN in which each node i has a representation $z_i \in \mathbb{R}^d$, as proposed in [61]. Briefly, for a target node i in the graph, the representations of its neighboring nodes $\{z_j \mid j \in \mathcal{N}(i)\}$ are first transformed via a learned linear transformation W. Then, these transformed representations are gathered with predetermined weights α, followed by a non-linear function σ (ReLU [99]). This layer-wise propagation can be written as:

$$\boldsymbol{z}_i^{(l+1)} = \sigma\left(\boldsymbol{z}_i^{(l)} + \sum_{j \in \mathcal{N}(i)} \alpha_{ij} W \boldsymbol{z}_j^{(l)}\right) \tag{3.4}$$

or equivalently we can collect node representations into a matrix $Z \in \mathbb{R}^{d \times Tn}$

$$\boldsymbol{z}_i^{(l+1)} = \sigma\left(W Z^{(l)} \boldsymbol{\alpha}_i\right) \tag{3.5}$$

for $\boldsymbol{\alpha}_i \in [0,1]^n$ with 0 entries for nodes not neighboring i and $\boldsymbol{\alpha}_{ii} = 1$. In conventional GCN, the connections in the graph are known and coefficient vector $\boldsymbol{\alpha}_i$ are preset based on the symmetrically normalized adjacency matrix of features.

In this paper, we extend the conventional GCN to an attentional version, which we refer to as aGCN, by learning to adjust $\boldsymbol{\alpha}$. To predict attention from node features, we learn a 2-layer MLP over concatenated node features and compute a softmax over the resulting scores. The the attention for node i is

$$u_{ij} = w_h^T \sigma(W_a[\boldsymbol{z}_i^{(l)}, \boldsymbol{z}_j^{(l)}]) \tag{3.6}$$

$$\boldsymbol{\alpha}_i = \text{softmax}(\boldsymbol{u}_i), \tag{3.7}$$

where w_h and W_a are learned parameters and $[\cdot,\cdot]$ is the concatenation operation. By definition, we set $\boldsymbol{\alpha}_{ii} = 1$ and $\boldsymbol{\alpha}_{ij} = 0 \; \forall j \notin \mathcal{N}(i)$. As attention is a function of node features, each iteration results in altered attentions which affects successive iterations.

aGCN for Scene Graph Generation. Recall that from the previous sections we have a set of N object regions and m relationships. From these, we construct a graph G with nodes corresponding to object and relationship proposals. We insert edges between relation nodes and their associated objects. We also add skip-connect edges directly between all object nodes. These connections allow information to flow directly between object nodes. Recent work has shown that reasoning about object correlation can improve detection performance [100]. We apply aGCN to this graph to update object and relationship representations based on global context.

Note that our graph captures a number of different types of connections (*i.e.*`object` $\leftrightarrow$ `relationship`, `relationship` $\leftrightarrow$ `subject` and `object` $\leftrightarrow$ `object`). In addition, the information flow across each connection may be asymmetric (the informativeness of `subject` on `relationship` might be quite different from `relationship` to `subject`). We learn different transformations for each type and ordering – denoting the linear transform

from node type a to node type b as W^{ab} with s=subjects, o=objects, and r=relationships. Using the same notation as in Eq. 3.5 and writing object and relationship features as Z^o and Z^r, we write the representation update for object nodes as

$$\boldsymbol{z}_i^o = \sigma(\overbrace{W^{\texttt{skip}} Z^o \boldsymbol{\alpha}^{\texttt{skip}}}^{\substack{\text{Message from} \\ \text{Other Objects}}} + \overbrace{W^{sr} Z^r \boldsymbol{\alpha}^{sr} + W^{or} Z^r \boldsymbol{\alpha}^{or}}^{\substack{\text{Messages from} \\ \text{Neighboring Relationships}}}) \tag{3.8}$$

with $\boldsymbol{\alpha}_{ii}^{\texttt{skip}}$=1 and similarly for relationship nodes as

$$\boldsymbol{z}_i^r = \sigma(\boldsymbol{z}_i^r + \underbrace{W^{rs} Z^o \boldsymbol{\alpha}^{rs} + W^{ro} Z^o \boldsymbol{\alpha}^{ro}}_{\text{Messages from Neighboring Objects}}). \tag{3.9}$$

where $\boldsymbol{\alpha}$ are computed at each iteration as in Eq. 3.7.

One open choice is how to initialize the object and relationship node representations z which could potentially be set to any intermediate feature representation or even the pre-softmax output corresponding to class labels. In practice, we run both a visual and semantic aGCN computation – one with visual features and the other using pre-softmax outputs. In this way, we can reason about both lower-level visual details (*i.e.*two people are likely talking if they are facing one another) as well as higher-level semantic co-occurrences (*i.e.*cars have wheels). Further, we set the attention in the semantic aGCN to be that of the visual aGCN – effectively modulating the flow of semantic information based on visual cues. This also enforces that real-world objects and relationships represented in both graphs interact with others in the same manner.

3.3.4 Loss Function

In Graph R-CNN, we factorize the scene graph generation process into three sub-processes: $P(\boldsymbol{R}, \boldsymbol{O}|\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{I})$, $P(\boldsymbol{E}|\boldsymbol{V}, \boldsymbol{I})$, $P(\boldsymbol{V}|\boldsymbol{I})$, which were described above. During training, each of these sub-processes are trained with supervision. For $P(\boldsymbol{V}|\boldsymbol{I})$, we use the same loss as used in RPN, which consists of a binary cross entropy loss on proposals and a regression

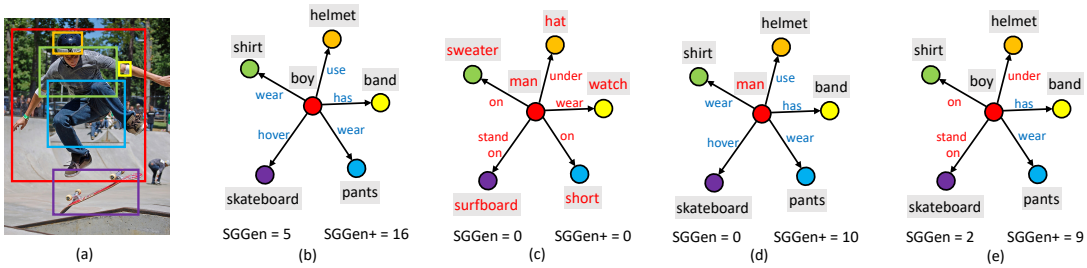


Figure 3.3: A example to demonstrate the difference between `SGGen` and `SGGen+`. Given the input image (a), its ground truth scene graph is depicted in (b). (c)-(e) are three generated scene graphs. For clarity, we merely show the connections with *boy*. At the bottom of each graph, we compare the number of correct predictions for two metrics.

loss for anchors. For $P(\boldsymbol{E}|\boldsymbol{V}, \boldsymbol{I})$, we use another binary cross entropy loss on the relation proposals. For the final scene graph generation $P(\boldsymbol{R}, \boldsymbol{O}|\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{I})$, two multi-class cross entropy losses are used for object classification and predicate classification.

3.4 Evaluating Scene Graph Generation

Scene graph generation is naturally a structured prediction problem over attributed graphs, and how to correctly and efficiently evaluate predictions is an under-examined problem in prior work on scene graph generation. We note that graph similarity based on minimum graph edit distance has been well-studied in graph theory [101]; however, computing exact solution is NP-complete and approximation APX-hard [102].

Prior work has circumvented these issues by evaluating scene graph generation under a simple triplet-recall based metric introduced in [82]. Under this metric which we will refer to as `SGGen`, the ground truth scene graph is represented as a set of {object, relationship, subject} triplets and recall is computed via exact match. That is to say, a triplet is considered 'matched' in a generated scene graph if all three elements have been correctly labeled, and both `object` and `subject` nodes have been properly localized (*i.e.*, bounding box IoU > 0.5). While simple to compute, this metric results in some unintuitive notions of similarity that we demonstrate in Fig. 3.3.

Fig. 3.3a shows an input image overlaid with bounding box localizations of correspond-

ingly colored nodes in the ground truth scene graph shown in (b). (c), (d), and (e) present erroneously labeled scene graphs corresponding to these same localizations. Even a casual examination of (c) and (d) yields the stark difference in their accuracy – while (d) has merely mislabeled the boy as a man, (c) has failed to accurately predict even a single node or relationship! Despite these differences, neither recalls a single complete triplet and are both scored identically under `SGGen` (*i.e.*, 0).

To address this issue, we propose a new metric called `SGGen+` as the augmentation of `SGGen`. `SGGen+` not only considers the triplets in the graph, but also the singletons (object and predicate). The computation of `SGGen+` can be formulated as:

$$Recall = \frac{C(O) + C(P) + C(T)}{N} \tag{3.10}$$

where $C(\cdot)$ is a counting operation, and hence $C(O)$ is the number of object nodes correctly localized and recognized; $C(P)$ is for predicate. Since the location of predicate depends on the location of subject and object, only if both subject and object are correctly localized and the predicate is correctly recognized, we will count it as one. $C(T)$ is for triplet, which is the same as `SGGen`. Here, N is the number of entries (the sum of number of objects, predicates and relationships) in the ground truth graph. In Fig. 3.3, using our `SGGen+`, the recall for graph (c) is still 0, since all predictions are wrong. However, the recall for graph (d) is not 0 anymore since most of the object and all predicate predictions are correct, except for one wrong prediction for the red node. Based on our new metric, we can obtain a much comprehensive measurement of scene graph similarity.

3.5 Experiments

Recently, there are some inconsistencies in existing work on scene graph generation in terms of data preprocessing, data split, and evaluation. This makes it difficult to systematically benchmark progress and cleanly compare numbers across papers. So we first clarify

the details of our experimental settings.

Datasets. There are a number of splits of the Visual Genome dataset that have been used in the scene graph generation literature [82, 95, 98]. The most commonly used is the one proposed in [82]. Hence, in our experiments, we follow their preprocessing strategy and dataset split. After preprocessing, the dataset is split into training and test sets, which contains 75,651 images and 32,422 images, respectively. In this dataset, the top-frequent 150 object classes and 50 relation classes are selected. Each image has around 11.5 objects and 6.2 relationships in the scene graph.

Training. For training, multiple strategies have been used in literature. In [82, 95, 96], the authors used two-stage training, where the object detector is pre-trained, followed by the joint training of the whole scene graph generation model. To be consistent with previous work [82, 95], we also adopt the two-stage training – we first train the object detector and then train the whole model jointly until convergence.

Metrics. We use four metrics for evaluating scene graph generation, including three previously used metrics and our proposed `SGGen+` metric:

- **Predicate Classification (PredCls)**: The performance for recognizing the relation between two objects given the ground truth locations.
- **Phrase Classification (PhrCls)**: The performance for recognizing two object categories and their relation given the ground truth locations.
- **Scene Graph Generation (SGGen)**: The performance for detecting objects (IoU $>$ 0.5) *and* recognizing the relations between object pairs.
- **Comprehensive Scene Graph Generation (SGGen+)**: Besides the triplets counted by `SGGen`, it considers the singletons and pairs (if any), as described earlier.

Evaluation. In our experiments, we multiply the classification scores for subjects, objects and their relationships, then sort them in descending order. Based on this order, we compute the recall at top 50 and top 100, respectively. Another difference in existing

literature in the evaluation protocol is w.r.t. the `PhrCls` and `PredCls` metrics. Some previous works [95, 96] used different models to evaluate along different metrics. However, such a comparison is unfair since the models could be trained to overfit the respective metrics. For meaningful evaluation, we evaluate a single model – the one obtained after joint training – across all metrics.

3.5.1 Implementation Details

We use Faster R-CNN [51] associated with VGG16 [103] as the backbone based on the PyTorch re-implementation [73]. During training, the number of proposals in RPN is 256. For each proposal, we perform ROI Align [52] pooling, to get a 7×7 response map, which is then fed to a two-layer MLP to obtain each proposal's representation. In RePN, the projection functions $\Phi(\cdot)$ and $\Psi(\cdot)$ are simply two-layer MLPs. During training, we sample 128 object pairs from the quadratic number of candidates. We then obtain the union of boxes of the two objects and extract a representation for the union. The threshold for box-pair NMS is 0.7. In aGCN, to obtain the attention for one node pair, we first project the object/predicate features into 256-d and then concatenate them into 512-d, which is then fed to a two-layer MLP with a 1-d output. For aGCN, we use two aGCN layers at the feature level and semantic level, respectively. The attention on the graph is updated in each aGCN layer at the feature level, which is fixed and sent to the aGCN at the semantic level.

Training. As mentioned, we perform stage-wise training – we first pretrain Faster R-CNN for object detection, and then fix the parameters in the backbone to train the scene graph generation model. SGD is used as the optimizer, with initial learning rate 1e-2 for both training stages.

3.5.2 Analysis on New Metric

We first quantitatively demonstrate the difference between our proposed metric `SGGen+` and `SGGen`. We compare them by perturbing ground truth scene graphs. We consider

Table 3.1: Comparisons between SGGen and SGGen+ under different perturbations.

Per. Type	n one	w/o relationship			w/ relationship			both		
Per. Ratio	0%	20%	50%	100%	20%	50%	100%	20%	50%	100%
SGGen	100.0	100.0	100.0	100.0	54.1	22.1	0.0	62.2	24.2	0.0
SGGen+	100.0	94.5	89.1	76.8	84.3	69.6	47.9	80.1	56.6	22.8

assigning random incorrect labels to objects; perturbing objects 1) without relationships, 2) with relationships, and 3) both. We vary the fraction of nodes which are perturbed among {20%, 50%, 100%}. Recall is reported for both metrics. As shown in Table 3.1, SGGen is completely insensitive to the perturbation of objects without relationships (staying at 100 consistently) since it only considers relationship triplets. Note that there are on average 50.1% objects without relationships in the dataset, which SGGen omits. On the other hand, SGGen is overly sensitive to label errors on objects with relationships (reporting 54.1 at only 20% perturbation where the overall scene graph is still quite accurate). Note that even at 100% perturbation the object localizations and relationships are still correct such that SGGen+ provides a non-zero score, unlike SGGen which considers the graph entirely wrong. Overall, we hope this analysis demonstrates that SCGen+ is more comprehensive compared to SCGen.

3.5.3 Quantitative Comparison

We compare our Graph R-CNN with recent proposed methods, including Iterative Message Passing (IMP) [82], Multi-level scene Description Network (MSDN) [95]. Furthermore, we evaluate the neural motif frequency baseline proposed in [97]. Note that previous methods often use slightly different pre-training procedures or data split or extra supervisions. For a fair comparison and to control for such orthogonal variations, we reimplemented IMP, MSDN and frequency baseline in our codebase. Then, we re-train IMP and MSDN based on our backbone – specifically, we used the same pre-trained object detector, and then jointly train the scene graph generator until convergence. We denote these as IMP† and

Table 3.2: Comparison on Visual Genome test set [81]. We reimplemented IMP [82] and MSDN [95] using the same object detection backbone for fair comparison.

	SGGen+		SGGen		PhrCls		PredCls	
Method	R@50	R@100	R@50	R@100	R@50	R@100	R@50	R@100
IMP [82]	-	-	3.4	4.2	21.7	24.4	44.8	53.0
MSDN [95]	-	-	7.7	10.5	19.3	21.8	63.1	66.4
Pixel2Graph [96]	-	-	9.7	11.3	26.5	30.0	68.0	75.2
IMP† [82]	25.6	27.7	6.4	8.0	20.6	22.4	40.8	45.2
MSDN† [95]	25.8	28.2	7.0	9.1	27.6	29.9	53.2	57.9
NM-Freq† [97]	26.4	27.8	6.9	9.1	23.8	27.2	41.8	48.8
Graph R-CNN (Us)	**28.5**	**35.9**	**11.4**	**13.7**	**29.6**	**31.6**	**54.2**	**59.1**

MSDN†. Using the same pre-trained object detector, we report the neural motif frequency baseline in [97] as NM-Freq†.

We report the scene graph generation performance in Table 3.2. The top three rows are numbers reported in the original paper, and the bottom four rows are the numbers from our re-implementations. First, we note that our re-implementations of IMP and MSDN (IMP† and MSDN†) result in performance that is close to or better than the originally reported numbers under some metrics (but not all), which establishes that the takeaway messages next are indeed due to our proposed architectural choices – relation proposal network and attentional GCNs. Next, we notice that Graph R-CNN outperforms IMP† and MSDN†. This indicates that our proposed Graph R-CNN model is more effective to extract the scene graph from images. Our approach also outperforms the frequency baseline on all metrics, demonstrating that our model has not just learned simple co-occurrence statistics from training data, but rather also captures context in individual images. More comprehensively, we compare with IMP and MSDN on the efficiency over training and inference. IMP uses 2.15$\times$ while MSDN uses 1.86$\times$ our method. During inference, IMP is 3.27$\times$ while MSDN is 3.80$\times$ slower than our Graph R-CNN. This is mainly due to the simplified architecture design (especially the aGCN for context propagation) in our model.

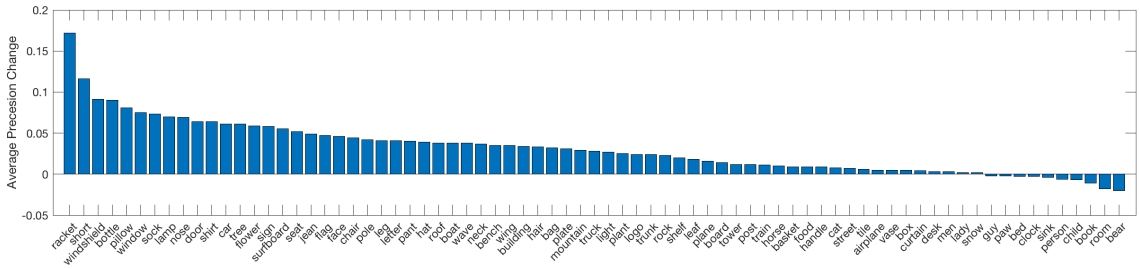


Figure 3.4: Per category object detection performance change after adding RePN.

Table 3.3: Ablation studies on Graph R-CNN. We report the performance based on four scene graph generation metrics and the object detection performance in mAP@0.5.

RePN	GCN	aGCN	Detection	SGGen+		SGGen		PhrCls		PredCls	
			mAP@0.5	R@50	R@100	R@50	R@100	R@50	R@100	R@50	R@100
-	-	-	20.4	25.9	27.9	6.1	7.9	17.8	19.9	33.5	38.4
✓	-	-	**23.6**	27.6	34.8	8.7	11.1	18.3	20.4	34.5	39.5
✓	✓	-	23.4	28.1	35.3	10.8	13.4	27.2	29.5	52.3	57.2
✓	-	✓	23.0	**28.5**	**35.9**	**11.4**	**13.7**	**29.4**	**31.6**	**54.2**	**59.1**

3.5.4 Ablation Study

In Graph R-CNN, we proposed two novel modules – relation proposal network (RePN) and attentional GCNs (aGCN). In this sub-section, we perform ablation studies to get a clear sense of how these two components affect the final performance. The left-most columns in Table 3.3 indicate whether or not we used RePN, GCN, and attentional GCN (aGCN) in our approach. The results are reported in the remaining columns of Table 3.3. We also report object detection performance mAP@0.5 following Pascal VOC's metric [104].

In Table 3.3, we find RePN boosts `SGGen` and `SGGen+` significantly. This indicates that our RePN can effectively prune the spurious connections between objects to achieve high recall for the correct relationships. We also notice it improves object detection significantly. In Fig. 3.4 we show the per category object detection performance change when RePN is added. For visual clarity, we dropped every other column when producing the plot. We can see that almost all object categories improve after adding RePN. Interestingly, we find the detection performance on categories like *racket*, *short*, *windshield*, *bottle* are

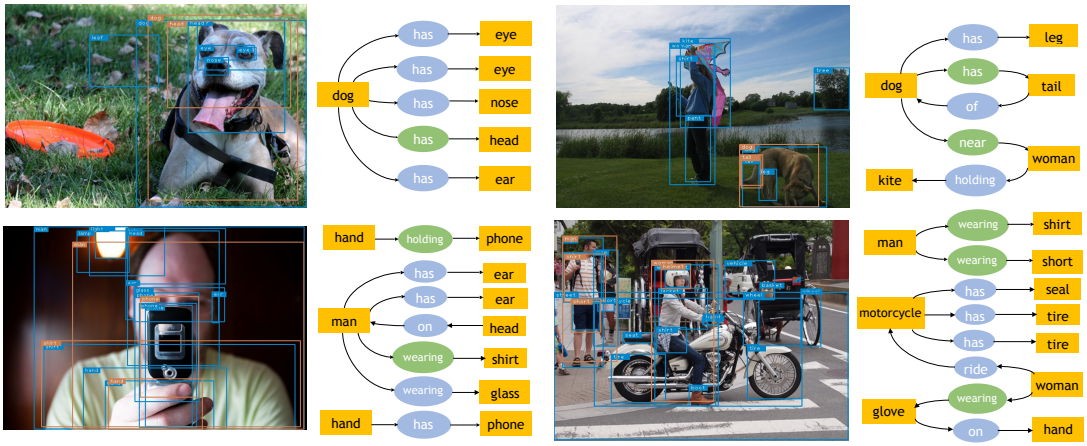


Figure 3.5: Qualitative results from Graph R-CNN. In images, blue and orange bounding boxes are ground truths and correct predictions, respectively. In scene graphs, blue ellipsoids are ground truth relationships while green ones denote correct predictions.

most significantly improved. Note that many of these classes are smaller objects that have strong relationships with other objects, e.g. rackets are often carried by people. Evaluating `PhrCls` and `PredCls` involves using the ground truth object locations. Since the number of objects in images (typically <25) is much less than the number of object proposals (64), the number of relation pairs is already very small. As a result, RePN has less effect on these two metrics.

By adding the aGCNs into our model, the performance is further improved. These improvements demonstrate that the aGCN in our Graph R-CNN can capture meaningful context across the graph. We also compare the performance of our model with and without attention. We see that by adding attention on top of GCNs, the performance is higher. This indicates that controlling the extent to which contextual information flows through the edges is important. These results align with our intuitions mentioned in the introduction. Fig. 3.5 shows generated scene graphs for test images. With RePN and aGCN, our model is able to generate higher recall scene graphs. The green ellipsoids shows the correct relationship predictions in the generated scene graph.

3.6 Discussion

In this chapter, we introduce a new model for scene graph generation – Graph R-CNN. Our model includes a relation proposal network (RePN) that efficiently and intelligently prunes out pairs of objects that are unlikely to be related, and an attentional graph convolutational network (aGCN) that effectively propagates contextual information across the graph. We also introdce a novel scene graph generation evaluation metric (`SGGen+`) that is more fine-grained and realistic than existing metrics. Our approach outperforms existing methods for scene graph generation, as evaluated using existing metrics and our proposed metric.

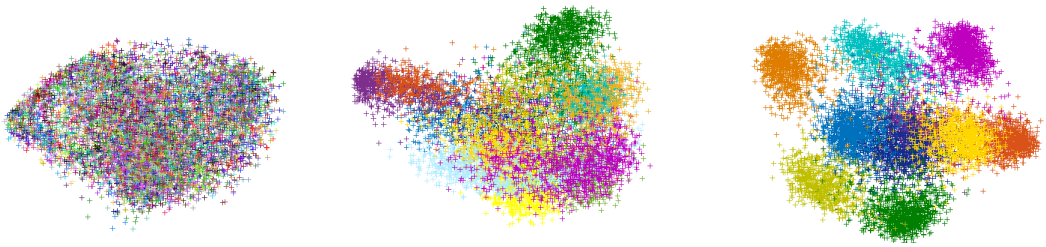
CHAPTER 4
LEVERAGE DATASET-LEVEL STRUCTURE FOR IMAGE CLUSTERING AND REPRESENTATION LEARNING

In this chapter, I will introduce how we can leverage the dataset-level structure for image clustering and representation learning. As we know, images are usually embed in a low-dimensional manifold. This manifold forms a structure in which some images are close to each other while some are far away from each other. Based on this observation, we propose a method to simultaneously cluster images and learn the representations from them. The intuition behind this is that, during the recovery of the image manifold, the structure itself provides us a good signal on which images are similar to others or the opposite. By building an affinity graph over an unlabeled image set, we can gradually evolve the graph by merging clusters and in turn update the representations for them. Extensive experiments demonstrate that the proposed method can not only cluster the images but also learn a good representation from them. We further show that the learned representations can be easily transferred and reused by other vision tasks.

4.1 Introduction

We are witnessing an explosion in visual content. Significant recent advances in machine learning and computer vision, especially via deep neural networks, have relied on supervised learning and availability of copious annotated data. However, manually labelling data is a time-consuming, laborious, and often expensive process. In order to make better use of unlabeled images, clustering and/or unsupervised learning is a promising direction.

In this work, we aim to address image clustering and representation learning on unlabeled images in a unified framework. It is a natural idea to leverage cluster ids of images as supervisory signals to learn representations and in turn the representations would be



(a) Initial stage (b) Middle stage (c) Final stage

Figure 4.1: Clustering outputs for MNIST [105] test set at different stages of the proposed method. We conduct PCA on the image representations and then choose the first three dimensions for visualization. Different colors correspond to different clusters. Samples are grouped together gradually and more discriminative representations are obtained.

beneficial to image clustering. At a high-level view, given a collection of n_s unlabeled images $\boldsymbol{I} = \{I_1, ..., I_{n_s}\}$, the global objective function for learning image representations and clusters can be written as:

$$\underset{\boldsymbol{y},\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{y}, \boldsymbol{\theta}|\boldsymbol{I}) \tag{4.1}$$

where $\mathcal{L}(\cdot)$ is a loss function, $\boldsymbol{y}$ denotes the cluster ids for all images, and $\boldsymbol{\theta}$ denotes the parameters for representations. If we hold one in $\{\boldsymbol{y}, \boldsymbol{\theta}\}$ to be fixed, the optimization can be decomposed into two alternating steps:

$$\underset{\boldsymbol{y}}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{y}|\boldsymbol{I}, \boldsymbol{\theta}) \tag{4.2a}$$

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{I}, \boldsymbol{y}) \tag{4.2b}$$

Intuitively, (4.2a) can be cast as a conventional clustering problem based on fixed representations, while (4.2b) is a standard supervised representation learning process.

In this paper, we propose an approach that alternates between the two steps – updating the cluster ids given the current representation parameters and updating the representation parameters given the current clustering result. Specifically, we cluster images using agglomerative clustering[106] and represent images via activations of a Convolutional Neural Network (CNN).

The reason to choose agglomerative clustering is three-fold: 1) it begins with an over-clustering, which is more reliable in the beginning when a good representation has not yet been learned. Intuitively, clustering with representations from a CNN initialized with random weights are not reliable, but nearest neighbors and over-clusterings are often acceptable; 2) These over-clusterings can be merged as better representations are learned; 3) Agglomerative clustering is a recurrent process and can naturally be interpreted in a recurrent framework.

Our final algorithm is farily intuitive. We start with an intial over-clustering, update CNN parameters (2b) using image cluster labels as supervisory signals, then merge clusters (2a) and iterate until we reach a stopping criterion. An outcome of the proposed framework is illustrated in Fig. 4.1. Initially, there are 1,762 clusters for MNIST test set (10k samples), and the representations (image intensities) are not that discriminative. After several iterations, we obtain 17 clusters and more discriminative representations. Finally, we obtain 10 clusters which are well-separated by the learned representations and interestingly correspond primarily to the groundtruth category labels in the dataset, even though the representation is learnt in an unsupervised manner. To summarize, the major contributions of our work are:

1 We propose a simple but effective end-to-end learning framework to jointly learn deep representations and image clusters from an unlabeled image set;

2 We formulate the joint learning in a recurrent framework, where merging operations of agglomerative clustering are expressed as a forward pass, and representation learning of CNN as a backward pass;

3 We derive *a single loss function* to guide agglomerative clustering and deep representation learning, which makes optimization over the two tasks seamless;

4 Our experimental results show that the proposed framework outperforms previous methods on image clustering and learns deep representations that can be transferred to other tasks and datasets.

4.2 Background

Clustering Clustering algorithms can be broadly categorized into hierarchical and partitional approaches [25]. Agglomerative clustering is a hierarchical clustering algorithm that begins with many small clusters, and then merges clusters gradually [106, 107, 108]. As for partitional clustering methods, the most well-known is K-means [109], which minimizes the sum of square errors between data points and their nearest cluster centers. Related ideas form the basis of a number of methods, such as expectation maximization (EM) [110, 111], spectral clustering [112, 113, 114], and non-negative matrix factorization (NMF) based clustering [115, 116, 117].

Deep Representation Learning Many works use raw image intensity or hand-crafted features [118, 119, 120, 121, 122, 123] combined with conventional clustering methods. Recently, representations learned using deep neural networks have presented significant improvements over hand-designed features on many computer vision tasks, such as image classification [75, 124, 103, 125], object detection [76, 10, 126, 51], etc. However, these approaches rely on supervised learning with large amounts of labeled data to learn rich representations. A number of works have focused on learning representations from unlabled image data. One class of approaches cater to reconstruction tasks, such as auto-encoders [127, 128, 129, 130, 131], deep belief networks (DBN) [132], etc. Another group of techniques learn discriminative representations after fabricating supervisory signals for images, and then finetune them supervisedly for downstream applications [133, 134, 135]. Unlike our approach, the fabricated supervisory signal in these previous works is not updated during representation learning.

Combination A number of works have explored combining image clustering with representation learning. In [136], the authors proposed to learn a non-linear embedding of the undirected affinity graph using stacked autoencoder, and then ran K-means in the embedding space to obtain clusters. In [137], a deep semi-NMF model was used to factorize the input into multiple stacking factors which are initialized and updated layer by layer. Us-

ing the representations on the top layer, K-means was implemented to get the final results. Unlike our work, they do not jointly optimize for the representation learning and clustering.

To connect image clustering and representation learning more closely, [138] conducted image clustering and codebook learning iteratively. However, they learned codebook over SIFT feature [8], and did not learn deep representations. Instead of using hand-crafted features, Chen [139] used DBN to learn representations, and then conducted a nonparametric maximum margin clustering upon the outputs of DBN. Afterwards, they fine-tuned the top layer of DBN based on clustering results. A more recent work on jointly optimizing two tasks is found in [140], where the authors trained a task-specific deep architecture for clustering. The deep architecture is composed of sparse coding modules which can be jointly trained through back propagation from a cluster-oriented loss. However, they used sparse coding to extract representations for images, while we use a CNN. Instead of fixing the number of clusters to be the number of categories and predicted labels based on softmax outputs, we predict the labels using agglomerative clustering based on the learned representations. In our experiments we show that our approach outperforms [140].

4.3 Approach

4.3.1 Notation

We denote an image set with n_s images by $\boldsymbol{I} = \{I_1, ..., I_{n_s}\}$. The cluster labels for this image set are $\boldsymbol{y} = \{y_1, ..., y_{n_s}\}$. $\boldsymbol{\theta}$ are the CNN parameters, based on which we obtain deep representations $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_{n_s}\}$ from $\boldsymbol{I}$. Given the predicted image cluster labels, we organize them into n_c clusters $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, ..., \mathcal{C}_{n_c}\}$, where $\mathcal{C}_i = \{\boldsymbol{x}_k | y_k = i,\ \forall k \in 1, ..., n_s\}$. $\mathcal{N}_i^{K_s}$ are the K_s nearest neighbours of $\boldsymbol{x}_i$, and $\mathcal{N}_{\mathcal{C}_i}^{K_c}$ is the set of K_c nearest neighbour clusters of $\mathcal{C}_i$. For convenience, we sort clusters in $\mathcal{N}_{\mathcal{C}_i}^{K_c}$ in descending order of affinity with $\mathcal{C}_i$ so that the nearest neighbour $\mathrm{argmax}_{C \in \boldsymbol{\mathcal{C}}^t} \boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C})$ is the first entry $\mathcal{N}_{\mathcal{C}_i}^{K_c}[1]$. Here, $\mathcal{A}$ is a function to measure the affinity (or similarity) between two clusters. We add a superscript t to $\{\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\mathcal{C}}\}$ to refer to their states at timestep t. We use $\boldsymbol{\mathcal{Y}}$ to denote the

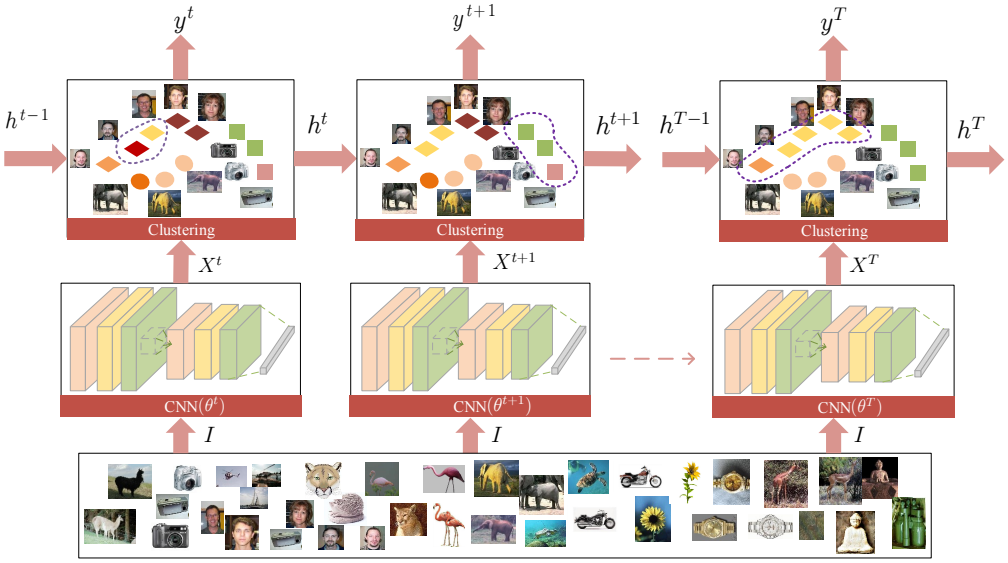


Figure 4.2: Proposed recurrent framework for unsupervised learning of deep representations and image clusters.

sequence $\{\boldsymbol{y}^1, ..., \boldsymbol{y}^T\}$ with T timesteps.

4.3.2 Agglomerative Clustering

As background, we first briefly describe conventional agglomerative clustering [106, 107]. The core idea in agglomerative clustering is to merge two clusters at each step until some stopping conditions. Mathematically, it tries to find two clusters $\mathcal{C}_a$ and $\mathcal{C}_b$ by

$$\{\mathcal{C}_a, \mathcal{C}_b\} = \underset{\mathcal{C}_i, \mathcal{C}_j \in \boldsymbol{\mathcal{C}}, i \neq j}{\operatorname{argmax}} \boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C}_j) \tag{4.3}$$

There are many methods to compute the affinity between two clusters [106, 107, 141, 142, 143]. More details can be found in [25]. We now describe how the affinity is measured by $\mathcal{A}$ in our approach.

4.3.3 Affinity Measure

First, we build a directed graph $G =< \mathcal{V}, \mathcal{E} >$, where $\mathcal{V}$ is the set of vertices corresponding to deep representations $\boldsymbol{X}$ for $\boldsymbol{I}$, and $\mathcal{E}$ is the set of edges connecting vertices. We define an affinity matrix $\boldsymbol{W} \in \mathbb{R}^{n_s \times n_s}$ corresponding to the edge set. The weight from vertex $\boldsymbol{x}_i$

to $\boldsymbol{x}_j$ is defined by

$$\boldsymbol{W}(i,j) = \begin{cases} exp(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2}{\sigma^2}), & \text{if } \boldsymbol{x}_j \in \mathcal{N}_i^{K_s} \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

where $\sigma^2 = \frac{a}{n_s K_s} \sum_{\boldsymbol{x}_i \in \boldsymbol{X}} \sum_{\boldsymbol{x}_j \in \mathcal{N}_i^{K_s}} ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2$. This way to build up a directed graph can be found in many previous works such as [142, 143]. Here, a and K_s are two predefined parameters (their values are listed in Table 4.2). After constructing a directed graph for samples, we then adopt the graph degree linkage in [143] to measure the affinity between cluster $\mathcal{C}_i$ and $\mathcal{C}_j$, denoted by $\mathcal{A}(\mathcal{C}_i, \mathcal{C}_j)$.

4.3.4 A Recurrent Framework

Our key insight is that agglomerative clustering can be interpreted as a recurrent process in the sense that it merges clusters over multiple timesteps. Based on this insight, we propose a recurrent framework to combine the image clustering and representation learning processes.

As shown in Fig. 4.2, at the timestep t, images $\boldsymbol{I}$ are first fed into the CNN to get representations $\boldsymbol{X}^t$ and then used in conjunction with previous hidden state $\boldsymbol{h}^{t-1}$ to predict current hidden state $\boldsymbol{h}^t$, i.e, the image cluster labels at timestep t. In our context, the output at timestep t is $\boldsymbol{y}^t = \boldsymbol{h}^t$. Hence, at timestep t

$$\boldsymbol{X}^t = f_r(\boldsymbol{I}|\boldsymbol{\theta}^t) \tag{4.5a}$$

$$\boldsymbol{h}^t = f_m(\boldsymbol{X}^t, \boldsymbol{h}^{t-1}) \tag{4.5b}$$

$$\boldsymbol{y}^t = f_o(\boldsymbol{h}^t) = \boldsymbol{h}^t \tag{4.5c}$$

where f_r is a function to extract deep representations $\boldsymbol{X}^t$ for input $\boldsymbol{I}$ using the CNN parameterized by $\boldsymbol{\theta}^t$, and f_m is a merging process for generating $\boldsymbol{h}^t$ based on $\boldsymbol{X}^t$ and $\boldsymbol{h}^{t-1}$.

In a typical Recurrent Neural Network, one would unroll all timesteps at each training iteration. In our case, that would involve performing agglomerative clustering until we obtain the desired number of clusters, and then update the CNN parameters by back-propagation.

In this work, we introduce a *partial unrolling* strategy, *i.e.*, we split the overall T timesteps into multiple periods, and unroll one period at a time. The intuitive reason we unroll partially is that the representation of the CNN at the beginning is not reliable. We need to update CNN parameters to obtain more discriminative representations for the following merging processes. In each period, we merge a number of clusters and update CNN parameters for a fixed number of iterations at the end of the period. An extreme case would be one timestep per period, but it involves updating the CNN parameters too frequently and is thus time-consuming. Therefore, the number of timesteps per period (and thus the number of clusters merged per period) is determined by a parameter in our approach. We elaborate on this more in Sec. 4.3.6.

4.3.5 Objective Function

In our recurrent framework, we accumulate the losses from all timesteps, which is formulated as

$$\mathcal{L}(\{\boldsymbol{y}^1, ..., \boldsymbol{y}^T\}, \{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^T\}|\boldsymbol{I}) = \sum_{t=1}^{T} \mathcal{L}^t(\boldsymbol{y}^t, \boldsymbol{\theta}^t|\boldsymbol{y}^{t-1}, \boldsymbol{I}) \tag{4.6}$$

Here, $\boldsymbol{y}^0$ takes each image as a cluster. At timestep t, we find two clusters to merge given $\boldsymbol{y}^{t-1}$. In conventional agglomerative clustering, the two clusters are determined by finding the maximal affinity over all pairs of clusters. In this paper, we introduce a criterion that considers not only the affinity between two clusters but also the local structure surrounding the clusters. Assume from $\boldsymbol{y}^{t-1}$ to $\boldsymbol{y}^t$, we merged a cluster $\mathcal{C}_i^t$ and its nearest neighbour. Then the loss at timestep t is a combination of negative affinities, that is,

$$\mathcal{L}^t(\boldsymbol{y}^t, \boldsymbol{\theta}^t|\boldsymbol{y}^{t-1}, \boldsymbol{I}) = -\boldsymbol{\mathcal{A}}(\mathcal{C}_i^t, \mathcal{N}_{\mathcal{C}_i^t}^{K_c}[1]) \tag{4.7a}$$

$$-\frac{\lambda}{(K_c-1)}\sum_{k=2}^{K_c}\Big(\boldsymbol{\mathcal{A}}(\mathcal{C}_i^t,\mathcal{N}_{\mathcal{C}_i^t}^{K_c}[1])-\boldsymbol{\mathcal{A}}(\mathcal{C}_i^t,\mathcal{N}_{\mathcal{C}_i^t}^{K_c}[k])\Big) \tag{4.7b}$$

where λ weighs (4.7a) and (4.7b). Note that $\boldsymbol{y}^t$, $\boldsymbol{y}^{t-1}$ and $\boldsymbol{\theta}^t$ are not explicitly presented at the right side, but they determine the loss via the image cluster labels and affinities among clusters. On the right side of the above equation, there are two terms: 1) (4.7a) measures the affinity between cluster $\mathcal{C}_i$ and its nearest neighbour, which follows conventional agglomerative clustering; 2) (4.7b) measures the difference between affinity of $\mathcal{C}_i$ to its nearest neighbour cluster and affinities of $\mathcal{C}_i$ to its other neighbour clusters. This term takes the local structure into account. See Sec. 4.3.5 for detailed explanation.

It is hard to simultaneously derive the optimal $\{\boldsymbol{y}^1,...,\boldsymbol{y}^T\}$ and $\{\boldsymbol{\theta}^1,...,\boldsymbol{\theta}^T\}$ that minimize the overall loss in Eq. (4.6). As aforementioned, we optimize iteratively in a recurrent process. We divide T timesteps into P partially unrolled periods. In each period, we fix $\boldsymbol{\theta}$ and search optimal $\boldsymbol{y}$ in the forward pass, and then in the backward pass we derive optimal $\boldsymbol{\theta}$ given the optimal $\boldsymbol{y}$. Details will be explained in the following sections.

Forward Pass

In forward pass of the p-th ($p \in \{1,...,P\}$) partially unrolled period, we update the cluster labels with $\boldsymbol{\theta}$ fixed to $\boldsymbol{\theta}^p$, and the overall loss in period p is

$$\mathcal{L}^p(\boldsymbol{\mathcal{Y}}^p|\boldsymbol{\theta}^p,\boldsymbol{I})=\sum_{t=t_p^s}^{t_p^e}\mathcal{L}^t(\boldsymbol{y}^t|\boldsymbol{\theta}^p,\boldsymbol{y}^{t-1},\boldsymbol{I}) \tag{4.8}$$

where $\boldsymbol{\mathcal{Y}}^p$ is the sequence of image labels in period p, and $[t_p^s,t_p^e]$ is the corresponding timesteps in period p. For optimization, we follow a greedy search similar to conventional agglomerative clustering. Starting from the time step t_p^s, it finds one cluster and its nearest neighbour to merge so that $\mathcal{L}^t$ is minimized over all possible cluster pairs.

In Fig. 4.3, we present a toy example to explain the reason why we employ the term (4.7b). As shown, it is often the case that the clusters are densely populated in some regions while sparse in some other regions. In conventional agglomerative clustering, it will choose

(a) (b)

Figure 4.3: A toy illustration of (a) conventional agglomerative clustering strategy and (b) the proposed one. For simplification, we use a single circle to represent a cluster/sample. In conventional agglomerative clustering, node b and its nearest neighbour are chosen to merge because they are closest to each other; while node e is chosen in our proposed strategy considering the local structure.

two clusters with largest affinity (or smallest loss) at each time no mater where the clusters are located. In this specific case, it will choose cluster $\mathcal{C}_b$ and its nearest neighbour to merge. In contrast, as shown in Fig. 4.3(b), our algorithm by adding (4.7b) will find cluster $\mathcal{C}_e$, because it is not only close to it nearest neighbour, but also relatively far away from its other neighbours, i.e., the local structure is considered around one cluster. Another merit of introducing (4.7b) is that it will allow us to write the loss in terms of triplets as explained next.

Backward Pass

In forward pass of the p-th partially unrolled period, we have merged a number of clusters. Let the sequence of optimal image cluster labels be given by $\boldsymbol{\mathcal{Y}}_*^p = \{\boldsymbol{y}_*^t\}$, and clusters merged in forward pass are denoted by $\{[\mathcal{C}_*^t, \mathcal{N}_{\mathcal{C}_*^t}^{K_c}[1]]\}$, $t \in \{t_p^s, ..., t_p^e\}$. In the backward pass, we aim to derive the optimal $\boldsymbol{\theta}$ to minimize the losses generated in forward pass. Because the clustering in current period is conditioned on the clustering results of all previous periods, we accumulate the losses of all p periods, i.e.,

$$\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{\mathcal{Y}}_*^1, ..., \boldsymbol{\mathcal{Y}}_*^p\}, \boldsymbol{I}) = \sum_{k=1}^{p} \mathcal{L}^k(\boldsymbol{\theta}|\boldsymbol{\mathcal{Y}}_*^k, \boldsymbol{I}) \tag{4.9}$$

Minimizing (4.9) w.r.t $\boldsymbol{\theta}$ leads to representation learning on $\boldsymbol{I}$ supervised by $\{\boldsymbol{\mathcal{Y}}_*^1, ..., \boldsymbol{\mathcal{Y}}_*^p\}$

or $\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^{t_p^e}\}$. Based on (4.7a) and (4.7b), the loss in Eq. 4.9 is reformulated to

$$-\frac{\lambda}{K_c - 1} \sum_{t=1}^{t_p^e} \sum_{k=2}^{K_c} \left(\lambda' \boldsymbol{\mathcal{A}}(\mathcal{C}_*^t, \mathcal{N}_{\mathcal{C}_*^t}^{K_c}[1]) - \boldsymbol{\mathcal{A}}(\mathcal{C}_*^t, \mathcal{N}_{\mathcal{C}_*^t}^{K_c}[k]) \right) \tag{4.10}$$

where $\lambda' = (1 + 1/\lambda)$. (4.10) is a loss defined on clusters of points, which needs the entire dataset to estimate, making it difficult to use batch-based optimization. However, we show that this loss can be approximated by a sample-based loss, enabling us to compute unbiased estimators for the gradients using batch-statistics.

The intuition behind reformulation of the loss is that agglomerative clustering starts with each datapoint as a cluster, and clusters at a higher level in the hierarchy are formed by merging lower level clusters. Thus, affinities between clusters can be expressed in terms of affinities between datapoints. We show in the supplement that the loss in (4.10) can be approximately reformulated as

$$\mathcal{L}(\boldsymbol{\theta} | \boldsymbol{y}_*^{t_p^e}, \boldsymbol{I}) = -\frac{\lambda}{K_c - 1} \sum_{i,j,k} (\gamma \boldsymbol{\mathcal{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \boldsymbol{\mathcal{A}}(\boldsymbol{x}_i, \boldsymbol{x}_k)) \tag{4.11}$$

where γ is a weight whose value depends on λ' and how clusters are merged during the forward pass. $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are from the same cluster, while $\boldsymbol{x}_k$ is from the neighbouring clusters, and their cluster labels are merely determined by the final clustering result $\boldsymbol{y}_*^{t_p^e}$. To further simplify the optimization, we instead search $\boldsymbol{x}_k$ in at most K_c neighbour samples of x_i from other clusters in a training batch. Hence, the batch-wise optimization can be performed using conventional stochastic gradient descent method. Note that such triplet losses have appeared in other works [144, 145]. Because it is associated with a weight, we call (A.24) the weighted triplet loss.

4.3.6 Optimization

Given an image dataset with n_s samples, we assume the number of desired clusters n_c^* is given to us as is standard in clustering. Then we can build up a recurrent process with $T = n_s - n_c^*$ timesteps, starting by regarding each sample as a cluster. However, such initialization makes the optimization time-consuming, especially when datasets contain a

Algorithm 1 Joint Optimization on $\boldsymbol{y}$ and $\boldsymbol{\theta}$

Input:

$\boldsymbol{I}$: = collection of image data;

n_c^*: = target number of clusters;

Output:

$\boldsymbol{y}^*, \boldsymbol{\theta}^*$: = final image labels and CNN parameters;

$t \leftarrow 0$; $p \leftarrow 0$

Initialize $\boldsymbol{\theta}$ and $\boldsymbol{y}$

repeat

  Update $\boldsymbol{y}^t$ to $\boldsymbol{y}^{t+1}$ by merging two clusters

  if $t = t_p^e$ **then**

    Update $\boldsymbol{\theta}^p$ to $\boldsymbol{\theta}^{p+1}$ by training CNN

    $p \leftarrow (p+1)$

  end if

  $t \leftarrow t+1$

until Cluster number reaches n_c^*

$\boldsymbol{y}^* \leftarrow \boldsymbol{y}^t$; $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}^p$ =0

large number of samples. To address this problem, we can first run a fast clustering algorithm to get the initial clusters. Here, we adopt the initialization algorithm proposed in [146] for fair comparison with their experiment results. Note that other kind of initializations can also be used, e.g. K-means. Based on the algorithm in [146], we obtain a number of clusters which contain a few samples for each (average is about 4 in our experiments). Given these initial clusters, our optimization algorithm learns deep representations and clusters. The algorithm is outlined in Alg. 1. In each partially unrolled period, we perform forward and backward passes to update $\boldsymbol{y}$ and $\boldsymbol{\theta}$, respectively. Specifically, in the forward pass, we merge two clusters at each timestep. In the backward pass, we run about 20 epochs to update $\boldsymbol{\theta}$, and the affinity matrix W is also updated based on the new representation. The duration of the p-th period is $n_p = ceil(\eta \times n_c^s)$ timesteps, where n_c^s is the number of clusters at the beginning of current period, and η is a parameter called *unrolling rate* to control the number of timesteps. The less η is, the more frequently we update $\boldsymbol{\theta}$.

4.4 Experiments

4.4.1 Image Clustering

We compare our approach with 12 clustering algorithms, including K-means [109], NJW spectral clustering (SC-NJW) [112], self-tuning spectral clustering (SC-ST)[113], large-scale spectral clustering (SC-LS) [147], agglomerative clustering with average linkage (AC-Link)[25], Zeta function based agglomerative clustering (AC-Zell) [142], graph degree linkage-based agglomerative clustering (AC-GDL) [143], agglomerative clustering via path integral (AC-PIC) [146], normalized cuts (N-Cuts) [114], locality preserving non-negative matrix factorization (NMF-LP) [116], NMF with deep model (NMF-D) [137], task-specific clustering with deep model (TSC-D) [140].

For evaluation, we use a commonly used metric: normalized mutual information (NMI) [148]. It ranges in $[0, 1]$. Larger value indicates more precise clustering results.

Table 4.1: Datasets used in our experiments.

Dataset	*MNIST*	*USPS*	*COIL20*	*COIL100*	*UMist*	*FRGC-v2.0*	*CMU-PIE*	*YTF*
#Samples	70000	11000	1440	7200	575	2462	2856	10000
#Categories	10	10	20	100	20	20	68	41
Image Size	28×28	16×16	128×128	128×128	112×92	32×32	32×32	55×55

Datasets. We evaluate the clustering performance on two hand-written digit image datasets (MNIST [105] and USPS[1]), two multi-view object image datasets (COIL20 and COIL100 [149]), and four face image datasets (UMist [150], FRGC-v2.0[2], CMU-PIE [151], Youtube-Face (YTF)) [152]. The number of samples and categories, and image size are listed in Table 4.1. MNIST consists of training set (60,000) and testing set (10,000). To compare with different approaches, we experiment on the full set (MNIST-full) and testing set (MNIST-test), separately. For face image datasets such as UMist, CMU-PIE, we use the images provided as is without any changes. For FRGC-v2.0 and YTF datasets, we first crop faces

[1] `http://www.cs.nyu.edu/~roweis/data.html`
[2] `http://www3.nd.edu/~cvrl/CVRL/Data_Sets.html`

and then resize them to a constant size. In FRGC-v2.0 dataset, we randomly choose 20 subjects. As for YTF dataset, we choose the first 41 subjects which are sorted by their names in alphabet order.

Table 4.2: Hyper-parameters in our approach.

Hyper-parameter	K_s	a	K_c	λ	γ	η
Value	20	1.0	5	1.0	2.0	0.9 or 0.2

Experimental Setup. All the hyper-parameters and their values for our approach are listed in Table 4.2. In our experiments, K_s is set to 20, the same value to [143]. a and λ are simply set to 1.0. We search the values of K_c and γ for best performance on MNIST-test set. The unrolling rate η for first four datasets is 0.9; and 0.2 for face datasets. The target cluster number n_c^* is set to be the number of categories in each dataset.

We use Caffe [153] to implement our approach. We stacked multiple combinations of convolutional layer, batch normalization layer, ReLU layer and pooling layer. For all the convolutional layers, the number of channels is 50, and filter size is 5×5 with stride = 1 and padding = 0. For pooling layer, its kernel size is 2 and stride is 2. To deal with varying image sizes across datasets, the number of stacked convolutional layers for each dataset is chosen so that the size of the output feature map is about 10×10. On the top of all CNNs, we append an inner product (*ip*) layer whose dimension is 160. *ip* layer is followed by a L2-normalization layer before being fed to the weighted triplet loss layer or used for clustering. For each partially unrolled period, the base learning rate is set to 0.01, momentum 0.9, and weight decay 5×10^{-5}. We use the *inverse* learning rate decay policy, with *Gamma*=0.0001 and *Power*=0.75. Stochastic gradient descent (SGD) is adopted for optimization.

Quantitative Comparison. We report NMI for different methods on various datasets. Results are averaged from 3 runs. We report the results by re-running the code released by

Table 4.3: Quantitative clustering performance (NMI) for different algorithms using image intensities as input.

Dataset	*COIL20*	*COIL100*	*USPS*	*MNIST-test*	*MNIST-full*	*UMist*	*FRGC*	*CMU-PIE*	*YTF*
K-means [109]	0.775	0.822	0.447	0.528	0.500	0.609	0.389	0.549	0.761
SC-NJW [112]	0.860	0.872	0.409	0.528	0.476	0.727	0.186	0.543	0.752
SC-ST [113]	0.673	0.706	0.342	0.756	0.416	0.611	0.431	0.581	0.620
SC-LS [147]	0.877	0.833	0.681	0.756	0.706	0.810	0.550	0.788	0.759
N-Cuts [114]	0.768	0.861	0.382	0.386	0.411	0.782	0.285	0.411	0.742
AC-Link [25]	0.512	0.711	0.579	0.662	0.686	0.643	0.168	0.545	0.738
AC-Zell [142]	0.954	0.963	0.774	0.810	0.017	0.755	0.351	0.910	0.733
AC-GDL [143]	0.945	0.954	0.854	0.864	0.017	0.755	0.351	0.934	0.622
AC-PIC [146]	0.950	0.964	0.840	0.853	0.017	0.750	0.415	0.902	0.697
NMF-LP [116]	0.720	0.783	0.435	0.467	0.452	0.560	0.346	0.491	0.720
NMF-D [137]	0.692	0.719	0.286	0.243	0.148	0.500	0.258	0.983/0.910	0.569
TSC-D [140]	0.928	-	-	-	0.651	-	-	-	-
OURS-SF	**1.000**	0.978	0.858	0.876	0.906	**0.880**	0.566	0.984	**0.848**
OURS-RC	**1.000**	**0.985**	**0.913**	**0.915**	**0.913**	0.877	**0.574**	**1.00**	**0.848**

Table 4.4: Quantitative clustering performance (NMI) for different algorithms using our learned representations as inputs.

Dataset	*COIL20*	*COIL100*	*USPS*	*MNIST-test*	*MNIST-full*	*UMist*	*FRGC*	*CMU-PIE*	*YTF*
K-means [109]	0.926	0.919	0.758	0.908	0.927	0.871	0.636	0.956	0.835
SC-NJW [112]	0.915	0.898	0.753	0.878	0.931	0.833	0.625	0.957	0.789
SC-ST [113]	0.959	0.922	0.741	0.911	0.906	0.847	**0.651**	0.938	0.741
SC-LS [147]	0.950	0.905	0.780	0.912	**0.932**	**0.879**	0.639	0.950	0.802
N-Cuts [114]	0.963	0.900	0.705	0.910	0.930	0.877	0.640	0.995	0.823
AC-Link [25]	0.896	0.884	0.783	0.901	0.918	0.872	0.621	0.990	0.803
AC-Zell [142]	**1.000**	0.989	0.910	0.893	0.919	0.870	0.551	**1.000**	0.821
AC-GDL [143]	**1.000**	0.985	0.913	**0.915**	0.913	0.870	0.574	**1.000**	**0.84**2
AC-PIC [146]	**1.000**	**0.990**	**0.914**	0.909	0.907	0.870	0.553	**1.000**	0.829
NMF-LP [116]	0.855	0.834	0.729	0.905	0.926	0.854	0.575	0.690	0.788

original papers. For those that did not release the code, the corresponding results are borrowed from the papers. We find the results we obtain are somewhat different from the one reported in original papers. We suspect that these differences may be caused by the different experimental settings or the released code is changed from the one used in the original paper. For all test algorithms, we conduct L2-normalization on the image intensities since it empirically improves the clustering performance. We report our own results in two cases: 1) the straight-forward clustering results obtained when the recurrent process finish, denoted by OURS-SF; 2) the clustering results obtained by re-running clustering algorithm after obtaining the final representation, denoted by OURS-RC. The quantitative results are

shown in Table 4.3. In the table cells, the value before '/' is obtained by re-running code while the value after '/' is that reported in previous papers.

As we can see from Table 4.3, both OURS-SF and OURS-RC outperform previous methods on all datasets with noticeable margin. Interestingly, we achieved perfect results (NMI = 1) on COIL20 and CMU-PIE datasets, which means that all samples in the same category are clustered into the same group. The agglomerative clustering algorithms, such as AC-Zell, AC-GDL and AC-PIC perform better than other algorithms generally. However, on MNIST-full test, they all perform poorly. The possible reason is that MNIST-full has 70k samples, and these methods cannot cope with such large-scale dataset when using image intensity as representation. However, this problem is addressed by our learned representation. We show that we achieved analogous performance on MNIST-full to MNIST-test set. In most cases, we can find OURS-RC performs better on datasets that have room for improvement. We believe the reason is that OURS-RC uses the final learned representation over the entire clustering process, while OURS-SF starts with image intensity, which indicates that the learned representation is more discriminative than image intensity. [3]

We further evaluate the performance of different algorithms based on clustering accuracy (AC) metric, as a supplement to the NMI metric used in our main paper. As we can see from table 4.5, the proposed method outperform other methods on all datasets, which has similar trend as evaluated using NMI. Meanwhile, according to table 4.6, all other clustering algorithms are boosted after using the learned representation as evaluated on AC. These results further prove the proposed method is superior to other clustering algorithms and also learns powerful deep representations that generalize well across different clustering algorithms.

Table 4.5: Quantitative clustering performance (AC) for different algorithms using image intensities as input.

Dataset	*COIL20*	*COIL100*	*USPS*	*MNIST-test*	*MNIST-full*	*UMist*	*FRGC*	*CMU-PIE*	*YTF*
K-means [109]	0.665	0.580	0.467	0.560	0.564	0.419	0.327	0.246	0.548
SC-NJW [112]	0.641	0.544	0.413	0.220	0.502	0.551	0.178	0.255	0.551
SC-ST [113]	0.417	0.300	0.308	0.454	0.311	0.411	0.358	0.293	0.290
SC-LS [147]	0.717	0.609	0.659	0.740	0.714	0.568	0.407	0.549	0.544
N-Cuts [114]	0.544	0.577	0.314	0.304	0.327	0.550	0.235	0.155	0.536
AC-Link [25]	0.251	0.269	0.421	0.693	0.657	0.398	0.175	0.201	0.547
AC-Zell [142]	0.867	0.811	0.575	0.693	0.112	0.517	0.266	0.765	0.519
AC-GDL [143]	0.865	0.797	0.867	0.933	0.113	0.563	0.266	0.842	0.430
AC-PIC [146]	0.855	0.840	0.855	0.920	0.115	0.576	0.320	0.797	0.472
NMF-LP [116]	0.621	0.553	0.522	0.479	0.471	0.365	0.259	0.229	0.546
OURS-SF	**1.000**	0.894	0.922	0.940	0.959	**0.809**	**0.461**	0.980	**0.684**
OURS-RC	**1.000**	**0.916**	**0.950**	**0.961**	**0.964**	**0.809**	**0.461**	**1.000**	**0.684**

Table 4.6: Quantitative clustering performance (AC) for different algorithms using our learned representations as inputs.

Dataset	*COIL20*	*COIL100*	*USPS*	*MNIST-test*	*MNIST-full*	*UMist*	*FRGC*	*CMU-PIE*	*YTF*
K-means [109]	0.821	0.751	0.776	0.957	0.969	0.761	0.476	0.834	0.660
SC-NJW [112]	0.738	0.659	0.716	0.868	0.972	0.707	0.485	0.776	0.521
SC-ST [113]	0.851	0.705	0.661	0.960	0.958	0.697	0.496	0.896	0.575
SC-LS [147]	0.867	0.735	0.792	0.960	**0.973**	0.733	0.502	0.802	0.571
N-Cuts [114]	0.888	0.626	0.634	0.959	0.971	**0.798**	**0.504**	0.981	0.441
AC-Link [25]	0.678	0.539	0.773	0.955	0.964	0.795	0.495	0.947	0.602
AC-Zell [142]	**1.000**	0.931	0.879	0.879	0.969	0.790	0.449	**1.000**	0.644
AC-GDL [143]	**1.000**	0.920	0.949	**0.961**	0.878	0.790	0.461	**1.000**	**0.677**
AC-PIC [146]	**1.000**	**0.950**	**0.955**	0.958	0.882	0.790	0.438	**1.000**	0.652
NMF-LP [116]	0.769	0.603	0.778	0.955	0.970	0.725	0.481	0.504	0.575

4.4.2 Robustness Analysis

We choose the two most important parameters: unfolding rate η and K_s for evaluating the robustness of our approach to variations in these parameters. In these experiments, we set all the other parameters except for the target one to default values listed in Table 2 in the main paper. As we can see from Fig. 4.4, when the unfolding rate increases, the performance is not affected much for most of the datasets. For K_s, the performance is stable when $K_s <= 50$ for all datasets. It drops with larger values of K_s for a few datasets.

[3]We experimented with hand-crafted features such as HOG, LBP, spatial pyramid on a subset of the datasets with some of the better clustering algorithms from Table 4.3, and found that they performed worse.

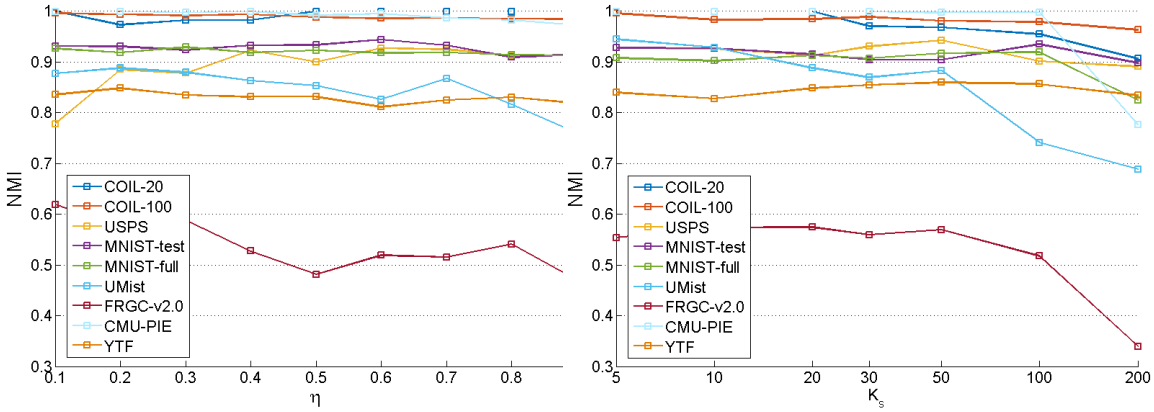


Figure 4.4: Clustering performance (NMI) with different η (left) and K_s (right).

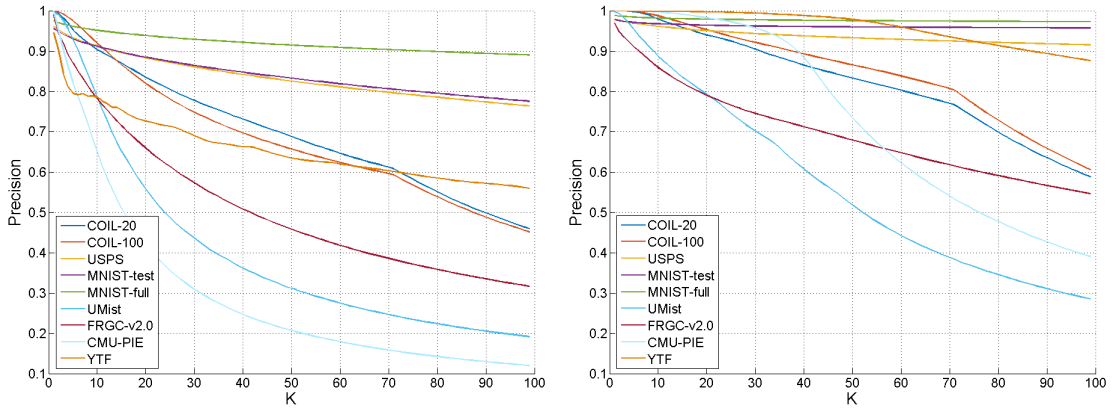


Figure 4.5: Average purity of K-nearest neighbour for varying values of K. Left is computed using raw image data, while right is computed using our learned representation.

Increasing K_s also result in similar degradation in the agglomerative clustering algorithms we compare to. This suggests that K_s should not be set to very large value in general.

4.4.3 Reliability Analysis

We evaluate the reliability by measuring the purity of samples at the beginning of our algorithm. Because we use agglomerative clustering, there are very few samples in each cluster at the beginning (average is about 4 in our experiments). Most samples in the same cluster tend to belong to the same category. Quantitatively, for each sample in a dataset, we count the number of samples (K_m) that belong to the same category within its K nearest neighbours, and then compute the precision K_m/K for it. In Fig. 4.5, we report the average

Table 4.7: Clustering performance (NMI) based on hand-crafted features.

Dataset	COIL100	MNIST-test	UMist	FRGC
SC-LS [147]	0.733↓	0.625↓	0.752↓	0.338↓
N-Cuts [114]	0.722↓	0.423↑	0.420↓	0.238↓
AC-PIC [146]	0.878↓	0.735↓	0.734↓	0.322↓

precision across all samples. As we can see, based on raw image data, all datasets have high ratios when K is smaller, and the ratios increase further when using our learned deep representations. Consequently, when K is small, the pseudo-labels are reliable enough to learn plausible deep representations.

4.4.4 Clustering based on Hand-crafted Features

We evaluate the performance of clustering based on image features, instead of image intensities. We choose three different types of datasets for testing: COIL100, MNIST-test and UMist, and three types of clustering algorithms including SC-LS [147], N-Cuts [114] and AC-PIC [146] for comparison since their better performance among all the algorithms. For these three datasets, we use spatial pyramid descriptor [154][4], histogram of oriented gradient (HOG) [7][5] and local binary pattern (LBP) [155] for representation, respectively. We report the results in Table 4.7. ↓ means performance become worse, and ↑ means it become better. Almost all algorithms perform worse than using original image as input. It indicates hand-crafted features should be designed dataset by dataset. In contrast, directly learning from image intensities achieves better performance.

4.4.5 Generalization Across Clustering Algorithms

We now evaluate if the representations learned by our joint agglomerative clustering and representation learning approach generalize to other clustering techniques. We re-run all the clustering algorithms without any changes of parameters, but using our learned deep

[4] `http://slazebni.cs.illinois.edu/research/SpatialPyramid.zip`
[5] `http://www.robots.ox.ac.uk/~vgg/research/caltech/phog.html`

Table 4.8: NMI performance across COIL20 and COIL100.

Layer	*data*	top(*ip*)	top-1	top-2
COIL20 → COIL100	0.924	0.927	**0.939**	0.934
COIL100 → COIL20	0.944	0.949	**0.957**	0.951

Table 4.9: NMI performance across MNIST-test and USPS.

Layer	*data*	top(*ip*)	top-1	top-2
MNIST-test → USPS	0.874	0.892	0.907	**0.908**
USPS → MNIST-test	0.872	0.873	**0.886**	-

representations as features. The results are shown in Table 4.4. It can be seen that all clustering algorithms obtain more precise image clusters by using our learned representation. Some algorithms like K-means, AC-Link that performed very poorly with raw intensities perform much better with our learned representations, and the variance in performance across all clustering algorithms is much lower. These results clearly demonstrate that our learned representation is not over-fitting to a single clustering algorithm, but generalizes well across various algorithms. Interestingly, using our learned representation, some of the clustering algorithms perform even better than AC-GDL we build on in our approach.

4.4.6 Transferring Learned Representation

Cross-Dataset Clustering

In this section, we study whether our learned representations generalize across datasets. We train a CNN based on our approach on one dataset, and then cluster images from another (but related) dataset using the image features extracted via the CNN. Specifically, we experiment on two dataset pairs: 1) multi-view object datasets (COIL20 and COIL100); 2) hand-written digit datasets (USPS and MNIST-test). We use the representation learned from one dataset to represent another dataset, followed by agglomerative clustering. Note that because the image sizes or channels are different across datasets, we resize the input images and/or expand the channels before feeding them to CNN. The experimental results

are shown in Table 4.8 and 4.9. We use the representations from top *ip* layer and also the *convolutional* or *pooling* layers (top-1, top-2) close to top layer for image clustering. In two tables, compared with directly using raw image from the *data* layer, the clustering performance based on learned representations from all layers improve, which indicates that the learned representations can be transferred across these datasets. As perhaps expected, the performance on target datasets is worse compared to learning on the target dataset directly. For COIL20 and COIL100, a possible reason is that they have different image categories. As for MNIST and USPS, the performance beats OURS-SF, but worse than OURS-RC. We find transferring representation learned on MNIST-test to USPS gets close performance to OURS-RC learned on USPS.

Face Verification

We now evaluate the performance of our approach by applying it to face verification. In particular, the representation is learned on Youtube-Face dataset and evaluated on LFW dataset [156] under the restricted protocol. For training, we randomly choose about 10k, 20k, 30k, 50k, 100k samples from YTF dataset. All these subsets have 1446 categories. We implement our approach to train CNN model and cluster images on the training set. Then, we remove the L2-normalization layer and append a softmax layer to fine-tune our unsupervised CNN model *based on the predicted image cluster labels*. Using the same training samples and CNN architecture, we also train a CNN model with a softmax loss supervised by the groundtruth labels of the training set. According to the evaluation protocol in [156], we run 10-fold cross-validation. The cosine similarity is used to compute the similarity between samples. In each of 10 cross-validations, nine folds are used to find the optimal threshold, and the remaining one fold is used for evaluation. The average accuracy is reported in Table. 4.10. As shown, though no groundtruth labels are used for representation learning in our approach, we obtain analogous performance to the supervised learning approach. Our approach even (slightly) beats the supervised learning method in one case.

Table 4.10: Face verification results on LFW.

#Samples	10k	20k	30k	50k	100k
Supervised	**0.737**	**0.746**	0.748	**0.764**	0.770
OURS	0.728	0.743	**0.750**	0.762	0.767

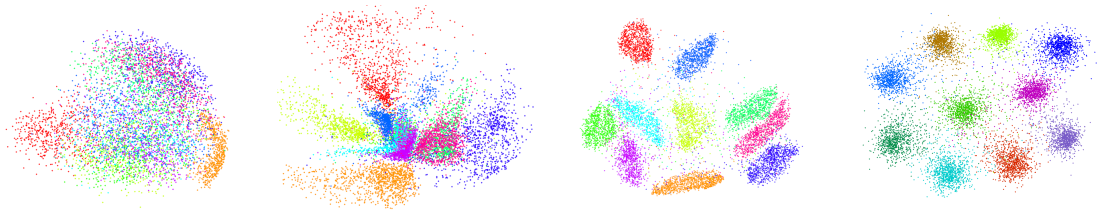
4.4.7 Image Classification

Recently, unsupervised representation learning is starting to achieve promising results for a variety of recognition tasks [157, 158, 159, 160]. We are interested in knowing whether the proposed method can also learn useful representation for image classification. We experiment with CIFAR-10 [68]. We follow the pipeline in [157], and base our experiments on their publicly available code. In this pipeline, codebook with 1600 codes is build upon 6×6 ZCA-whitened image patches, and then used to code the training and testing samples by extracting 1,600-d feature from each of 4 image quadrants. Afterwards, a linear SVM [161] is applied for image classification on 6,400-d feature. In our approach, the only difference is that we learn a new representation from 6×6 patches, and use these new representations to build the codebook with 1,600 codes. The CNN architecture we use contains two convolutional layers, each of which is combined with a ReLu and a pooling layer, followed by an inner product layer. Both convolutional layers have 50 3×3 filters with pad = 1. The kernel size of pooling layer is 2 with stride = 2. To save training time, 40k randomly extracted patches are extracted from 50k training set and used in both methods.

Classification accuracies on test set with different settings are shown in Table 4.11. We vary the number of training samples and evaluate the performance for representations from different layers. As we can see, the combination of representations from the first and second convolutional layer achieve the best performance. We also use the representation output by inner product layer to learn the codebook. However, it performs poorly. A possible reason is that it discards spatial information of image patches, which may be important for learning a codebook. When using 400k randomly extracted patches to learn the codebook, [157] achieved 77.9%. However, it is still lower than what we achieved. This performance

Table 4.11: Image classification accuracy on CIFAR-10.

#Samples	K-means [157]	conv1	conv2	conv1&2
5k	62.81%	63.05%	63.10%	**63.50%**
10k	68.01%	68.30%	68.46%	**69.11%**
25k	74.01%	72.83%	72.93%	**75.11%**
50k (full set)	76.59%	74.68%	74.68%	**78.55%**



(a) PCA. (b) Autoencoder. (c) Parametric t-SNE. (d) Our method.

Figure 4.6: Visualization of 10,000 MNIST test samples in different embedding spaces.

also beats several other methods listed in [158, 162, 159, 160].

4.4.8 Visualizing Data in Low Dimension

Projecting high-dimensional data into low-dimensional space can help people to intuitively understand the data. Though the proposed method is aimed to learn deep representations, we note that it can be naturally converted to a parametric visualization method by slightly alternating the objective. Instead of updating the affinities among samples based on the learned representations gradually, we consistently use the affinities among raw image data to perform the agglomerative cluster, which then guides representation learning in low-dimensional space. By this way, we can obtain a low-dimensional space (2D or 3D) which can retain the structure of the original data.

We compare three dimension reduction techniques, principle component analysis (PCA) [163], neighbourhood components analysis (NCA) [164], and parametric t-SNE [165]. Though both [165] and our visualization method are based on neural networks, there are two main differences: 1) In [165], a Kullback-Leibler divergence between the joint distributions of original data and the embedded data is considered. However, in our method,

we employ a weighted triplet loss that directly takes the local structure of embedded data into account; 2) In [165], the authors need to pretrain a stack of RBMs layer-by-layer, and then fine-tune the neural network. Nevertheless, we directly train the neural network from scratch end-to-end.

We perform experiments on MNIST dataset. In MNIST, 60,000 training samples are used to learn the low-dimensional embedding space, and 10,000 test samples are used for evaluation. To train a D-dimensional embedding, we first remove the normalization layer and then stack on the top another linear layer whose dimension is D. To thoroughly explore the local structure in the original data, we merge the clusters with a lower unfolding rate ($\eta = 0.2$). The learning process is stopped when the number of clusters reaches to 10. Though we stop the learning process as such, it should be noted that the stop criterion is not confined. For quantitative analysis, we compute the nearest-neighbor classification error and trustworthiness as in [165].

In Table 4.12, we show the 1-nearest neighbour classification error on MNIST test dataset. We copy the best results of the compared methods from [165]. As we can see, our method outperforms all three other methods across three different embedding dimensions. These results illustrates that our method can obtain low-dimensional embedding with better generalization ability.

For visualization, it is important to retain the original data structure in the embedding space. For quantitative comparison, we report the trustworthiness of learned low-dimensional embedding in Table 4.13. Larger value means better preservation of original data structure. As we can see, our method is not as good as parametric t-SNE. These results are explainable. During training, we merely pay attention to the local structure among samples from different clusters, while omitting the relations among samples within one cluster. Therefore, the algorithm will learn embeddings that discriminate clusters well but possibly disorder the samples in each cluster. We believe this can be solved by introducing a loss to confine the within-cluster structure. We leave this as a future work for limited space.

Table 4.12: 1-nearest neighbor classification error on MNIST dataset.

Method	2D	10D	30D
PCA [163]	0.782	0.430	0.108
NCA [164]	0.568	0.088	0.073
Autoencoder [128]	0.668	0.063	**0.027**
Param. t-SNE [165]	0.099	0.046	**0.027**
OURS	**0.067**	**0.019**	**0.027**

Table 4.13: Trustworthiness T(12) on MNIST dataset.

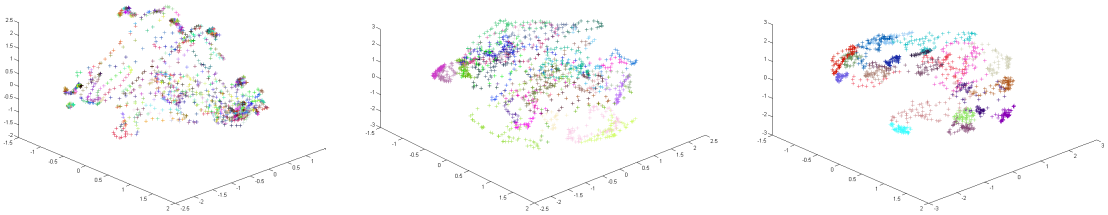
Method	2D	10D	30D
PCA [163]	0.744	0.991	0.998
NCA [164]	0.721	0.968	0.971
Autoencoder [128]	0.729	0.996	**0.999**
Param. t-SNE [165]	**0.927**	**0.997**	**0.999**
Ours	0.768	0.936	0.975

4.4.9 Visualizing Learned Representations

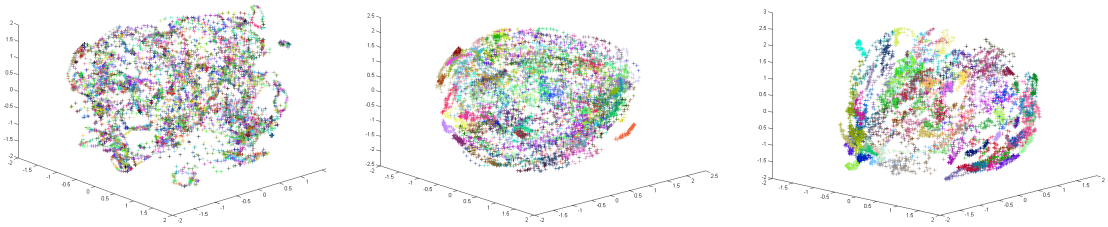
We show the first three principle components of learned representations in Fig. 4.7 and Fig. 4.8 at different stages. For comparison, we show the image intensities at the first column. We use different colors for representing different clusters that we predict during the algorithm. At the bottom of each plot, we give the number of clusters at the corresponding stage. At the final stage, the number of cluster is same to the number of categories in the dataset. After a number of iterations, we can learn more discriminative representations for the datasets, and thus facilitate more precise clustering results.

4.5 Discussion

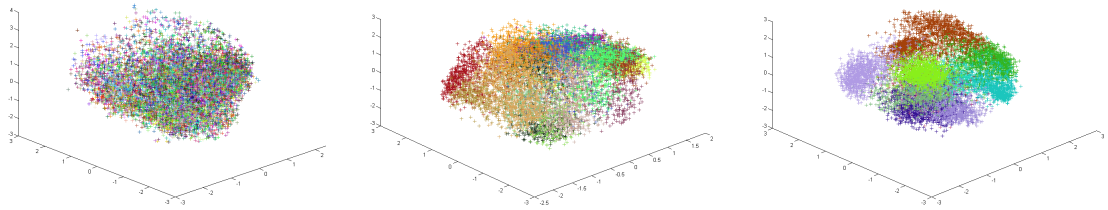
In this chapter, we have proposed an approach to jointly learn deep representations and image clusters. In our approach, we combined agglomerative clustering with CNNs and formulate them as a recurrent process. We used a partially unrolling strategy to divide the timesteps into multiple periods. In each period, we merged clusters step by step during the forward pass and learned representation in the backward pass, which are guided by a single weighted triplet-loss function. The extensive experiments on image clustering, deep representation transfer learning and image classification demonstrate that our approach can obtain more precise image clusters and discriminative representations that generalize well across many datasets and tasks.



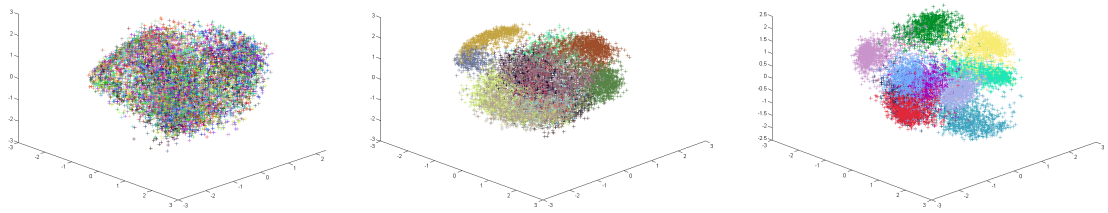
(a) Initial stage (421) (b) Middle stage (42) (c) Final stage (20)



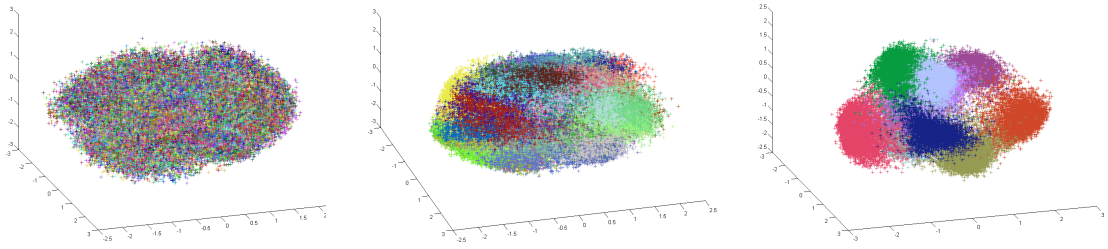
(d) Initial stage (2162) (e) Middle stage (216) (f) Final stage (100)



(g) Initial stage (2232) (h) Middle stage (22) (i) Final stage (10)

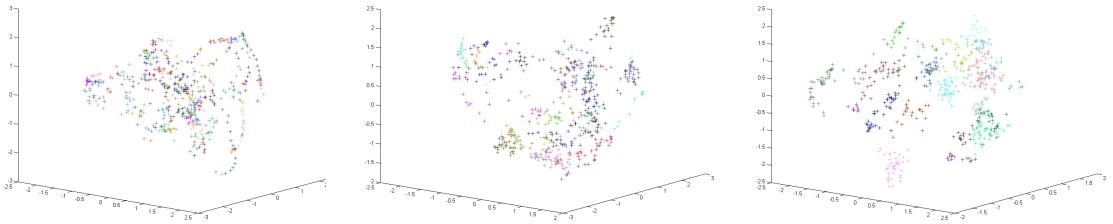


(j) Initial stage (1762) (k) Middle stage (22) (l) Final stage (10)

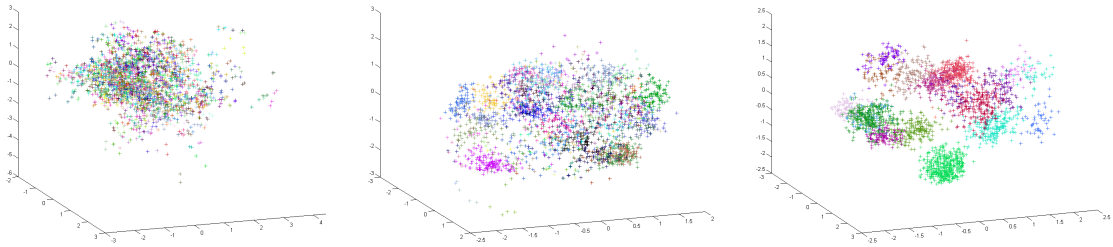


(m) Initial stage (11521) (n) Middle stage (115) (o) Final stage (10)

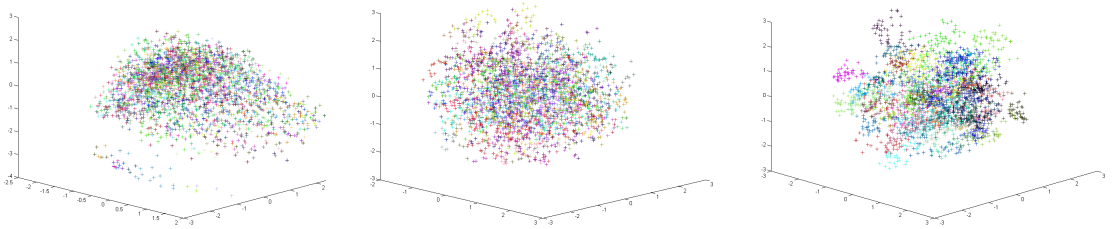
Figure 4.7: Learned representations at different stages on five datasets. From top to bottom, they are *COIL20*, *COIL100*, *USPS* and *MNIST-test* and *MNIST-full*. The first column are image intensities. For *MNIST-test*, we show another view point different from Fig.1.



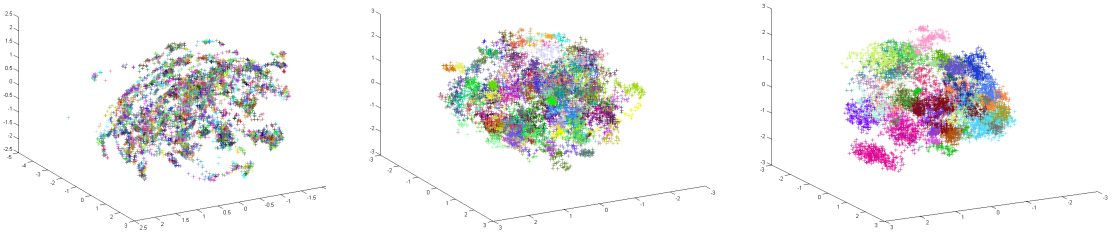
(a) Initial stage (188) (b) Middle stage (60) (c) Final stage (20)



(d) Initial stage (775) (e) Middle stage (128) (f) Final stage (20)



(g) Initial stage (775) (h) Middle stage (200) (i) Final stage (68)



(j) Initial stage (2814) (k) Middle stage (300) (l) Final stage (41)

Figure 4.8: Learned representations as different stages on four datasets. From top to bottom, they are *UMist*, *FRGC*, *CMU-PIE* and *YTF*. The first column are image intensities.

Part II

Structured Visual Generation

CHAPTER 5

LEVERAGE IMAGE-LEVEL STRUCTURE FOR IMAGE GENERATION

In this chapter, I will introduce how we can leverage the image-level structure for image generation. Recently, image generation techniques based on Generative Adversarial Netowrks (GAN) have draw a lot of attentions from the academia. Despite the striking performance for generating some specific categories of images, such as faces, recent models still face difficulties to generate realistic images with multiple objects in as image. In our work, we account for the structure prior in natural images and propose a layered-recursive image generation scheme, which explicitly generate background and foreground objects, and then put them together to generate the final images. Without any extra supervisions, our proposed method can clearly decompose the generation into separate background and foreground generation steps. We also propose two new metrics called Adversarial Accuracy and Adversarial Divergence to measure the quality of generated images. On both these two new metrics and the Inception Score metric, it is shown that our method outperforms one-shot image generation method with a large margin.

5.1 Introduction

Generative adversarial networks (GANs) [166] have shown significant promise as generative models for natural images. A flurry of recent work has proposed improvements over the original GAN work for image generation [167, 168, 169, 170, 171, 172], multi-stage image generation including part-based models [173, 174], image generation conditioned on input text or attributes [175, 176, 177], image generation based on 3D structure [178], and even video generation [179].

While the holistic 'gist' of images generated by these approaches is beginning to look natural, there is clearly a long way to go. For instance, the foreground objects in these

images tend to be deformed, blended into the background, and not look realistic or recognizable.

One fundamental limitation of these methods is that they attempt to generate images without taking into account that images are 2D projections of a 3D visual world, which has a lot of structures in it. This manifests as structure in the 2D images that capture this world. One example of this structure is that images tend to have a background, and foreground objects are placed in this background in contextually relevant ways.

We develop a GAN model that explicitly encodes this structure. Our proposed model generates images in a recursive fashion: it first generates a background, and then conditioned on the background generates a foreground along with a shape (mask) and a pose (affine transformation) that together define how the background and foreground should be composed to obtain a complete image. Conditioned on this composite image, a second foreground and an associated shape and pose are generated, and so on. As a byproduct in the course of recursive image generation, our approach generates some object-shape foreground-background masks in a completely unsupervised way, without access to *any* object masks for training. Note that decomposing a scene into foreground-background layers is a classical ill-posed problem in computer vision. By explicitly factorizing appearance and transformation, LR-GAN encodes natural priors about the images that the same foreground can be 'pasted' to the different backgrounds, under different affine transformations. According to the experiments, the absence of these priors result in degenerate foreground-background decompositions, and thus also degenerate final composite images.

We mainly evaluate our approach on four datasets: MNIST-ONE (one digit) and MNIST-TWO (two digits) synthesized from MNIST [181], CIFAR-10 [182] and CUB-200 [180]. We show qualitatively (via samples) and quantitatively (via evaluation metrics and human studies on Amazon Mechanical Turk) that LR-GAN generates images that globally look natural *and* contain clear background and object structures in them that are realistic and recognizable by humans as semantic entities. An experimental snapshot on CUB-200 is

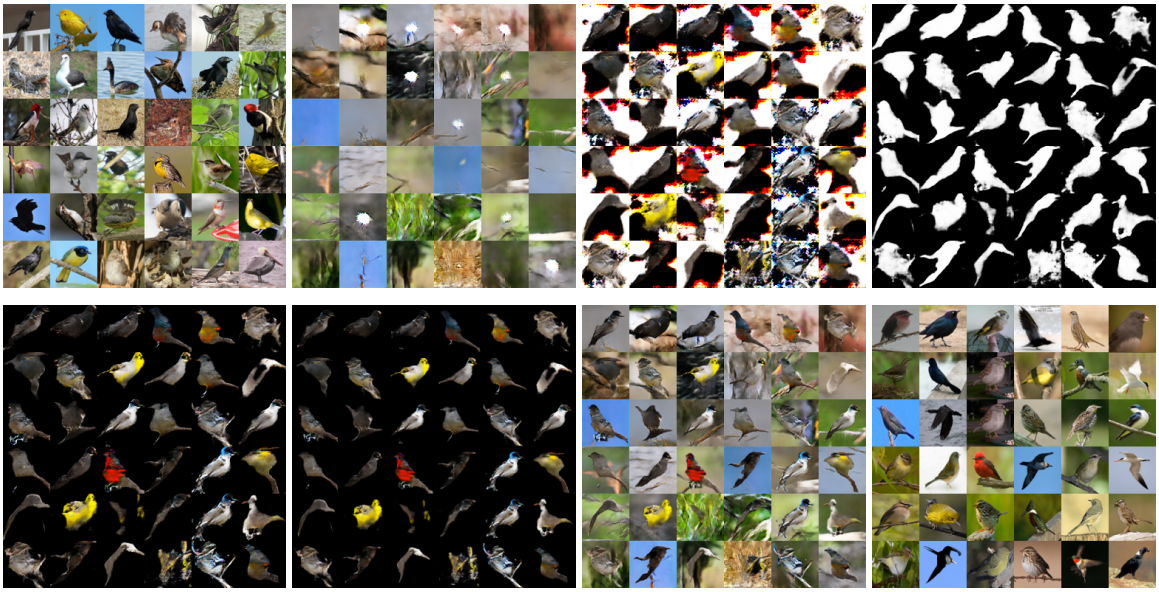


Figure 5.1: Generation results of our model on CUB-200 [180]. It generates images in two timesteps. At the first timestep, it generates background images, while generates foreground images, masks and transformations at the second timestep. Then, they are composed to obtain the final images. From top left to bottom right (row major), the blocks are real images, generated background images, foreground images, foreground masks, carved foreground images, carved and transformed foreground images, final composite images, and their nearest neighbor real images in the training set. Note that the model is trained in a completely unsupervised manner.

shown in Fig. 5.1. We also find that LR-GAN generates foreground objects that are contextually relevant to the backgrounds (e.g., horses on grass, airplanes in skies, ships in water, cars on streets, etc.). For quantitative comparison, besides existing metrics in the literature, we propose two new quantitative metrics to evaluate the quality of generated images. The proposed metrics are derived from the sufficient conditions for the closeness between generated image distribution and real image distribution, and thus supplement existing metrics.

5.2 Background

Early work in parametric texture synthesis was based on a set of hand-crafted features [183]. Recent improvements in image generation using deep neural networks mainly fall into one of the two stochastic models: variational autoencoders (VAEs) [27] and generative adversarial networks (GANs) [166]. VAEs pair a top-down probabilistic generative

network with a bottom up recognition network for amortized probabilistic inference. Two networks are jointly trained to maximize a variational lower bound on the data likelihood. GANs consist of a generator and a discriminator in a minmax game with the generator aiming to fool the discriminator with its samples with the latter aiming to not get fooled.

Sequential models have been pivotal for improved image generation using variational autoencoders: DRAW [184] uses attention based recurrence conditioning on the canvas drawn so far. In [185], a recurrent generative model that draws one object at a time to the canvas was used as the decoder in VAE. These methods are yet to show scalability to natural images. Early compelling results using GANs used sequential coarse-to-fine multiscale generation and class-conditioning [168]. Since then, improved training schemes [169] and better convolutional structure [167] have improved the generation results using GANs. PixelRNN [186] is also recently proposed to sequentially generates a pixel at a time, along the two spatial dimensions.

In this paper, we combine the merits of sequential generation with the flexibility of GANs. Our model for sequential generation imbibes a recursive structure that more naturally mimics image composition by inferring three components: appearance, shape, and pose. One closely related work combining recursive structure with GAN is that of [173] but it does not explicitly model object composition and follows a similar paradigm as by [184]. Another closely related work is that of [174]. It combines recursive structure and alpha blending. However, our work differs in three main ways: (1) We explicitly use a generator for modeling the foreground poses. That provides significant advantage for natural images, as shown by our ablation studies; (2) Our shape generator is separate from the appearance generator. This factored representation allows more flexibility in the generated scenes; (3) Our recursive framework generates subsequent objects conditioned on the current and previous hidden vectors, *and* previously generated object. This allows for explicit contextual modeling among generated elements in the scene. See Fig. 5.13 for contextually relevant foregrounds generated for the same background, or Fig. 5.6 for meaningful placement of

two MNIST digits relative to each.

Models that provide supervision to image generation using conditioning variables have also been proposed: Style/Structure GANs [178] learns separate generative models for style and structure that are then composed to obtain final images. In [177], GAN based image generation is conditioned on text and the region in the image where the text manifests, specified during training via keypoints or bounding boxes. While not the focus of our work, the model proposed in this paper can be easily extended to take into account these forms of supervision.

5.3 Preliminaries

5.3.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) consist of a generator G and a discriminator D that are simultaneously trained with competing goals: The generator G is trained to generate samples that can 'fool' a discriminator D, while the discriminator is trained to classify its inputs as either real (coming from the training dataset) or fake (coming from the samples of G). This competition leads to a minmax formulation with a value function:

$$\min_{\theta_G} \max_{\theta_D} \Big(\mathrm{E}_{\boldsymbol{x} \sim p_{data}(x)} [\log(D(\boldsymbol{x}; \theta_D))] + \mathrm{E}_{\boldsymbol{z} \sim p_z(z)} [\log(1 - D(G(\boldsymbol{z}; \theta_G); \theta_D))] \Big), \quad (5.1)$$

where $\boldsymbol{z}$ is a random vector from a standard multivariate Gaussian or a uniform distribution $p_z(z)$, $G(\boldsymbol{z}; \theta_G)$ maps $\boldsymbol{z}$ to the data space, $D(\boldsymbol{x})$ is the probability that $\boldsymbol{x}$ is real estimated by D. The advantage of the GANs formulation is that it lacks an explicit loss function and instead uses the discriminator to optimize the generative model. The discriminator, in turn, only cares whether the sample it receives is on the data manifold, and not whether it exactly matches a particular training example (as opposed to losses such as MSE). Hence, the discriminator provides a gradient signal *only* when the generated samples do not lie on the data manifold so that the generator can readjust its parameters accordingly. This form

of training enables learning the data manifold of the training set and not just optimizing to reconstruct the dataset, as in autoencoder and its variants.

While the GANs framework is largely agnostic to the choice of G and D, it is clear that generative models with the 'right' inductive biases will be more effective in learning from the gradient information [168, 173, 184, 177, 187]. With this motivation, we propose a generator that models image generation via a recurrent process – in each time step of the recurrence, an object with its own appearance and shape is generated and warped according to a generated pose to compose an image in layers.

5.3.2 Layered structure of image

An image taken of our 3D world typically contains a layered structure. One way of representing an image layer is by its appearance and shape. As an example, an image $\boldsymbol{x}$ with two layers, foreground $\boldsymbol{f}$ and background $\boldsymbol{b}$ may be factorized as:

$$\boldsymbol{x} = \boldsymbol{f} \odot \boldsymbol{m} + \boldsymbol{b} \odot (1 - \boldsymbol{m}), \tag{5.2}$$

where $\boldsymbol{m}$ is the mask depicting the shapes of image layers, and $\odot$ the element wise multiplication operator. Some existing methods assume the access to the shape of the object either during training [188] or *both* at train and test time [177, 187]. Representing images in layered structure is even straightforward for video with moving objects [189, 190, 191]. [179] generates videos by separately generating a fixed background and moving foregrounds. A similar way of generating single image can be found in [174].

Another way is modeling the layered structure with object appearance and pose as:

$$\boldsymbol{x} = ST(\boldsymbol{f}, \boldsymbol{a}) + \boldsymbol{b}, \tag{5.3}$$

where $\boldsymbol{f}$ and $\boldsymbol{b}$ are foreground and background, respectively; $\boldsymbol{a}$ is the affine transformation; ST is the spatial transformation operator. Several works fall into this group [192, 193, 185].

In [193], images are decomposed into layers of objects with specific poses in a variational autoencoder framework, while the number of objects (i.e., layers) is adaptively estimated in [185].

To contrast with these works, LR-GAN uses a layered composition, and the foreground layers simultaneously model all three dominant factors of variation: appearance $\boldsymbol{f}$, shape $\boldsymbol{m}$ and pose $\boldsymbol{a}$. We will elaborate it in the following section.

5.4 Layered Recursive GAN (LR-GAN)

The basic structure of LR-GAN is similar to GAN: it consists of a discriminator and a generator that are simultaneously trained using the minmax formulation of GAN, as described in §.5.3.1. The key innovation of our work is the layered recursive generator, which is what we describe in this section.

The generator in LR-GAN is recursive in that the image is constructed recursively using a recurrent network. Layered in that each recursive step composes an object layer that is 'pasted' on the image generated so far. Object layer at timestep t is parameterized by the following three constituents – 'canonical' appearance $\boldsymbol{f}_t$, shape (or mask) $\boldsymbol{m}_t$, and pose (or affine transformation) $\boldsymbol{a}_t$ for warping the object before pasting in the image composition.

Fig. 5.2 shows the architecture of the LR-GAN with the generator architecture unrolled for generating background $\boldsymbol{x}_0$ ($\doteq \boldsymbol{x}_b$) and foreground $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. At each time step t, the generator composes the next image $\boldsymbol{x}_t$ via the following recursive computation:

$$\boldsymbol{x}_t = \underbrace{ST(\boldsymbol{m}_t, \boldsymbol{a}_t)}_{\text{transformed mask}} \odot \underbrace{ST(\boldsymbol{f}_t, \boldsymbol{a}_t)}_{\text{transformed appearance}} + \underbrace{(1 - ST(\boldsymbol{m}_t, \boldsymbol{a}_t)) \odot \boldsymbol{x}_{t-1}}_{\text{pasting on image composed so far}}, \quad \forall t \in [1, T] \tag{5.4}$$

where $ST(\diamond, \boldsymbol{a}_t)$ is a spatial transformation operator that outputs the affine transformed version of $\diamond$ with $\boldsymbol{a}_t$ indicating parameters of the affine transformation.

Since our proposed model has an explicit transformation variable $\boldsymbol{a}_t$ that is used to warp the object, it can learn a canonical object representation that can be re-used to generate

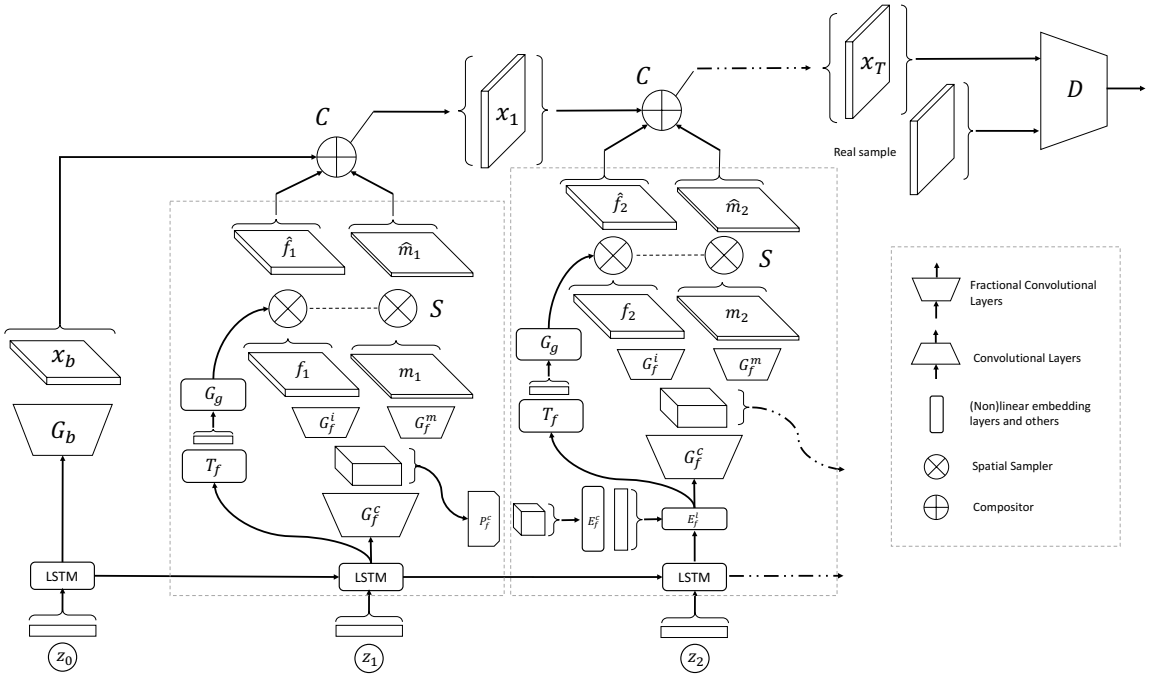


Figure 5.2: LR-GAN architecture unfolded to three timesteps. It mainly consists of one background generator, one foreground generator, temporal connections and one discriminator. The meaning of each component is explained in the legend.

scenes where the object occurs as mere transformations of it, such as different scales or rotations. By factorizing the appearance, shape and pose, the object generator can focus on separately capturing regularities in these three factors that constitute an object. We will demonstrate in our experiments that removing these factorizations from the model leads to its spending capacity in variability that may not solely be about the object in Section 5.5.2 and 5.5.3.

5.4.1 Details of Generator architecture

Fig. 5.2 shows our LR-GAN architecture in detail – we use different shapes to indicate different kinds of layers (convolutional, fractional convolutional, (non)linear, etc), as indicated by the legend. Our model consists of two main pieces – a background generator G_b and a foreground generator G_f. G_b and G_f do not share parameters with each other. G_b computation happens only once, while G_f is recurrent over time, i.e., all object gen-

erators share the same parameters. In the following, we will introduce each module and connections between them.

Temporal Connections. LR-GAN has two kinds of temporal connections – informally speaking, one on 'top' and one on 'bottom'. The 'top' connections perform the act of sequentially 'pasting' object layers (Eqn. 5.4). The 'bottom' connections are constructed by a LSTM on the noise vectors $\boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2$. Intuitively, this noise-vector-LSTM provides information to the foreground generator about what else has been generated in past. Besides, when generating multiple objects, we use a pooling layer P_f^c and a fully-connected layer E_f^c to extract the information from previous generated object response map. By this way, the model is able to 'see' previously generated objects.

Background Generator. The background generator G_b is purposely kept simple. It takes the hidden state of noise-vector-LSTM h_l^0 as the input and passes it to a number of fractional convolutional layers (also called 'deconvolution' layer in some papers) to generate images at its end. The output of background generator $\boldsymbol{x}_b$ will be used as the canvas for the following generated foregrounds.

Foreground Generator. The foreground generator G_f is used to generate an object with appearance and shape. Correspondingly, G_f consists of three sub-modules, G_f^c, which is a common 'trunk' whose outputs are shared by G_f^i and G_f^m. G_f^i is used to generate the foreground appearance $\boldsymbol{f}_t$, while G_f^m generates the mask $\boldsymbol{m}_t$ for the foreground. All three sub-modules consists of one or more fractional convolutional layers combined with batch-normalization and nonlinear layers. The generated foreground appearance and mask have the same spatial size as the background. The top of G_f^m is a sigmoid layer in order to generate one channel mask whose values range in $(0, 1)$.

Spatial Transformer. To spatially transform foreground objects, we need to estimate the transformation matrix. As in [194], we predict the affine transformation matrix with a linear layer T_f that has six-dimensional outputs. Then based on the predicted transformation matrix, we use a grid generator G_g to generate the corresponding sampling coordinates

in the input for each location at the output. The generated foreground appearance and mask share the same transformation matrix, and thus the same sampling grid. Given the grid, the sampler S will simultaneously sample the $\boldsymbol{f}_t$ and $\boldsymbol{m}_t$ to obtain $\hat{\boldsymbol{f}}_t$ and $\hat{\boldsymbol{m}}_t$, respectively. Different from [194], our sampler here normally performs downsampling, since the the foreground typically has smaller size than the background. Pixels in $\hat{\boldsymbol{f}}_t$ and $\hat{\boldsymbol{m}}_t$ that are from outside the extent of $\boldsymbol{f}_t$ and $\boldsymbol{m}_t$ are set to zero. Finally, $\hat{\boldsymbol{f}}_t$ and $\hat{\boldsymbol{m}}_t$ are sent to the compositor C which combines the canvas $\boldsymbol{x}_{t-1}$ and $\hat{\boldsymbol{f}}_t$ through layered composition with blending weights given by $\hat{\boldsymbol{m}}_t$ (Eqn. 5.4). Pseudo-code for our approach and detailed model configuration are provided below.

Algorithm 2 Stochastic Layered Recursive Image Generation

$\boldsymbol{z}_0 \sim \mathcal{N}(0, I)$
$\boldsymbol{x}_0 = G_b(\boldsymbol{z}_0)$, $\boldsymbol{h}_l^0 \leftarrow \mathbf{0}$, $\boldsymbol{c}_l^0 \leftarrow \mathbf{0}$
for $t \in [1 \cdots T]$ **do**
  $\boldsymbol{z}_t \sim \mathcal{N}(0, I)$
  $\boldsymbol{h}_l^t$, $\boldsymbol{c}_l^t \leftarrow$ LSTM([$\boldsymbol{z}_t$,$\boldsymbol{h}_l^{t-1}$, $\boldsymbol{c}_l^{t-1}$])
  if t = 1 **then**
    $\boldsymbol{y}_t \leftarrow \boldsymbol{h}_l^t$
  else
    $\boldsymbol{y}_t \leftarrow E_f^l([\boldsymbol{h}_l^t\ \boldsymbol{h}_f^{t-1}])$
  end if
  $\boldsymbol{s}_t \leftarrow G_f^c(\boldsymbol{y}_t)$, $a_t \leftarrow T_f(\boldsymbol{y}_t)$
  $\boldsymbol{f}_t \leftarrow G_f^i(\boldsymbol{s}_t)$, $\boldsymbol{m}_t \leftarrow G_f^m(\boldsymbol{s}_t)$
  $\boldsymbol{h}_f^t \leftarrow E_f^c \circ P_f^c(\boldsymbol{s}_t)$
  $\boldsymbol{x}_t \leftarrow ST(\boldsymbol{m}_t, \boldsymbol{a}_t) \odot ST(\boldsymbol{f}_t, \boldsymbol{a}_t) + (1 - ST(\boldsymbol{m}_t, \boldsymbol{a}_t)) \odot \boldsymbol{x}_{t-1}$
end for=0

5.4.2 New Evaluation Metrics

Several metrics have been proposed to evaluate GANs, such as Gaussian parzen window [166], Generative Adversarial Metric (GAM) [173] and Inception Score [169]. The common goal is to measure the similarity between the generated data distribution $P_g(\boldsymbol{x}) = G(\boldsymbol{z}; \theta_z)$ and the real data distribution $P(\boldsymbol{x})$. Most recently, Inception Score has been used in several works [169, 172]. However, it is an assymetric metric and could be easily fooled by generating centers of data modes.

In addition to these metrics, we present two new metrics based on the following intuition – a sufficient (but not necessary) condition for closeness of $P_g(\boldsymbol{x})$ and $P(\boldsymbol{x})$ is closeness of $P_g(\boldsymbol{x}|y)$ and $P(\boldsymbol{x}|y)$, i.e., distributions of generated data and real data conditioned on all possible variables of interest y, e.g., category label. One way to obtain this variable of interest y is via human annotation. Specifically, given the data sampled from $P_g(\boldsymbol{x})$ and $P(\boldsymbol{x})$, we ask people to label the category of the samples according to some rules. Note that such human annotation is often easier than comparing samples from the two distributions (e.g., because there is no 1:1 correspondence between samples to conduct forced-choice tests).

After the annotations, we need to verify whether the two distributions are similar in each category. Clearly, directly comparing the distributions $P_g(\boldsymbol{x}|y)$ and $P(\boldsymbol{x}|y)$ may be as difficult as comparing $P_g(\boldsymbol{x})$ and $P(\boldsymbol{x})$. Fortunately, we can use Bayes rule and alternatively compare $P_g(y|\boldsymbol{x})$ and $P(y|\boldsymbol{x})$, which is a much easier task. In this case, we can simply train a discriminative model on the samples from $P_g(\boldsymbol{x})$ and $P(\boldsymbol{x})$ together with the human annotations about categories of these samples. With a slight abuse of notation, we use $P_g(y|\boldsymbol{x})$ and $P(y|\boldsymbol{x})$ to denote probability outputs from these two classifiers (trained on generated samples vs trained on real samples). We can then use these two classifiers to compute the following two evaluation metrics:

Adversarial Accuracy: Computes the classification accuracies achieved by these two classifiers on a validation set, which can be the training set or another set of real images sampled from $P(\boldsymbol{x})$. If $P_g(\boldsymbol{x})$ is close to $P(\boldsymbol{x})$, we expect to see similar accuracies.

Adversarial Divergence: Computes the KL divergence between $P_g(y|\boldsymbol{x})$ and $P(y|\boldsymbol{x})$. The lower the adversarial divergence, the closer two distributions are. The low bound for this metric is exactly zero, which means $P_g(y|\boldsymbol{x}) = P(y|\boldsymbol{x})$ for all samples in the validation set.

As discussed above, we need human efforts to label the real and generated samples. Fortunately, we can further simplify this. Based on the labels given on training data, we

split the training data into categories, and train one generator for each category. With all these generators, we generate samples of all categories. This strategy will be used in our experiments on the datasets with labels given.

5.5 Experiment

We conduct qualitative and quantitative evaluations on three datasets: 1) MNIST [181]; 2) CIFAR-10 [182]; 3) CUB-200 [180]. To add variability to the MNIST images, we randomly scale (factor of 0.8 to 1.2) and rotate ($-\frac{\pi}{4}$ to $\frac{\pi}{4}$) the digits and then stitch them to 48×48 uniform backgrounds with random grayscale value between [0, 200]. Images are then rescaled back to 32×32. Each image thus has a different background grayscale value and a different transformed digit as foreground. We rename this sythensized dataset as **MNIST-ONE** (single digit on a gray background). We also synthesize a dataset **MNIST-TWO** containing two digits on a grayscale background. We randomly select two images of digits and perform similar transformations as described above, and put one on the left and the other on the right side of a 78×78 gray background. We resize the whole image to 64×64.

We develop LR-GAN based on open source code[1]. We assume the number of objects is known. Therefore, for MNIST-ONE, MNIST-TWO, CIFAR-10, and CUB-200, our model has two, three, two, and two timesteps, respectively. Since the size of foreground object should be smaller than that of canvas, we set the minimal allowed scale [2] in affine transforamtion to be 1.2 for all datasets except for MNIST-TWO, which is set to 2 (objects are smaller in MNIST-TWO). In LR-GAN, the background generator and foreground generator have similar architectures. One difference is that the number of channels in the background generator is half of the one in the foreground generator. We compare our results to that of DCGAN [167]. Note that LR-GAN without LSTM at the first timestep corresponds

[1] https://github.com/soumith/dcgan.torch

[2] Scale corresponds to the size of the target canvas with respect to the object – the larger the scale, the larger the canvas, and the smaller the relative size of the object in the canvas. 1 means the same size as the canvas.

Table 5.1: Information and model configurations on different datasets.

Dataset	MNIST-ONE	MNIST-TWO	CIFAR-10	CUB-200
Image Size	32	64	32	64
#Images	60,000	60,000	50,000	5,994
#Timesteps	2	3	2	2
#Parameters	5.25M/4.11M	7.53M/6.33M	5.26M/4.11M	27.3M/6.34M

exactly to the DCGAN. This allows us to run controlled experiments. In both generator and discriminator, all the (fractional) convolutional layers have 4×4 filter size with stride 2. As a result, the number of layers in the generator and discriminator automatically adapt to the size of training images. Table 5.1 lists the information and model configuration for different datasets. The dimensions of random vectors and hidden vectors are all set to 100. We also compare the number of parameters in DCGAN and LR-GAN. The numbers before '/' are our model, after '/' are DCGAN.

We use three metrics for quantitative evaluation, including Inception Score [169] and the proposed Adversarial Accuracy, Adversarial Divergence. Note that we report two versions of Inception Score. One is based on the pre-trained Inception net, and the other one is based on the pre-trained classifier on the target datasets. Algo. 2 illustrates the generative process in our model. $g(\star)$ evaluates the function g at $\star$. $\circ$ is a composition operator that composes its operands so that $f \circ g(\star) = f(g(\star))$.

5.5.1 Main Results

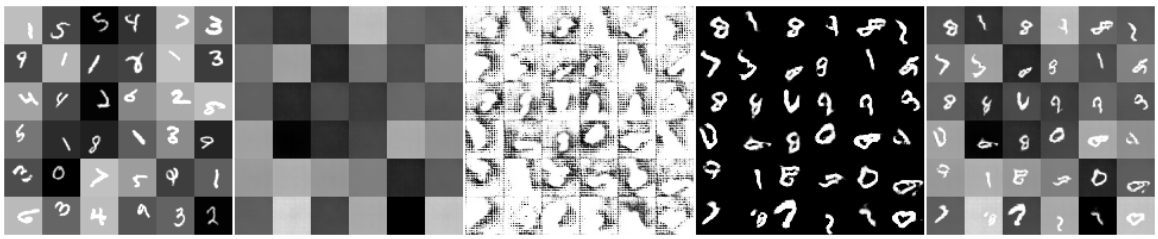


Figure 5.3: Generation results of our model on MNIST-ONE. From left to right, the image blocks are real images, generated background images, generated foreground images, generated masks and final composite images, respectively.

MNIST-ONE and MNIST-TWO. We first report the results on MNIST-ONE and MNIST-TWO. Fig. 5.3 shows the generation results of our model on MNIST-ONE. As we can see, our model generates the background and the foreground in separate timestep, and can disentagle the foreground digits from background nearly perfectly. Though initial values of the mask randomly distribute in the range of (0, 1), after training, the masks are nearly binary and accurately carve out the digits from the generated foreground. We further conduct human studies on generation results on MNIST-ONE. Specifically, we generate 1,000 images using both LR-GAN and DCGAN. As references, we also include 1000 real images. Then we ask the users on AMT to label each image to be one of the digits (0-9). We also provide them an option 'non recognizable' in case the generated image does not seem to contain a digit. Each image was judged by 5 unique workers. Similar to CIFAR-10, if an image is recognized to be the same digit by all 5 users, it is assigned to quality level 5. If it is not recognizable according to all users, it is assigned to quality level 0. Fig. 5.4 (left) shows the number of images assigned to all six quality levels. Compared to DCGAN, our model generated more samples with high quality levels. As expected, the real images have many samples with high quality levels. In Fig. 5.4 (right), we show the number of images that are recognized to each digit category (0-9). For qualitative comparison, we show examplar images at each quality level in Fig. 5.5. From left to right, the quality level increases from 0 to 5. As expected, the images with higher quality level are more clear.

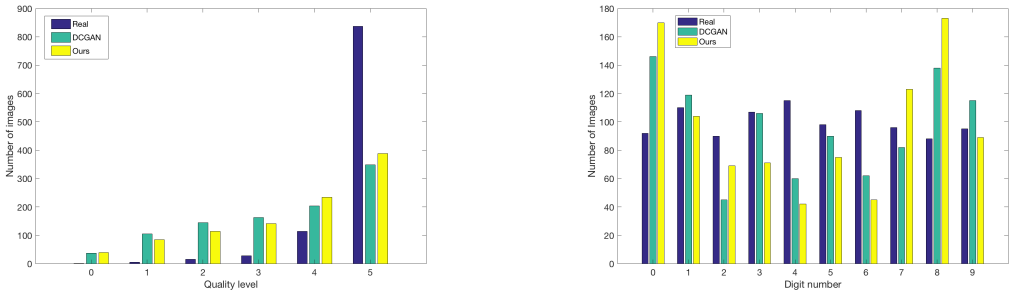


Figure 5.4: Statistics of annotations in human studies on MNIST-ONE. Left: distribution of quality level; Right: distribution of recognized digit categories.

Figure 5.5: Qualitative comparison on MNIST-ONE. Top three rows are samples generated by DCGAN. Bottom three rows are samples generated by LR-GAN. The quality level increases from left to right as determined via human studies.

Table 5.2: Quantitative comparison on MNIST-ONE.

Training Data	Real Images	DCGAN	Ours
Inception Score†	1.83±0.01	2.03±0.01	2.06±0.01
Inception Score††	9.15±0.04	6.42±0.03	7.15±0.04
Adversarial Accuracy	95.22 ± 0.25	26.12 ± 0.07	26.61 ± 0.06
Adversarial Divergence Score	0	8.47 ± 0.03	8.39 ± 0.04

†Evaluate using the pre-trained Inception net as [169]
††Evaluate using the supervisedly trained classifier based on the discriminator in LR-GAN.

For quantitative evaluation, we use the same way as for CIFAR-10. The classifier model used for contextual Inception Score is trained based on the training set. We generate 60,000 samples based on DCGAN and LR-GAN for evaluation, respectively. To obtain the Adversarial Accuracy and Adversarial Divergence, we first train 10 generators for 10 digit categories separately, and then use the generated samples to train the classifier. As shown in Table 5.2, our model has higher scores than DCGAN on both standard and contextual Inception Score. Also, our model has a slightly higher adversarial accuracy, and lower adversarial divergence than DCGAN. We find that the all three image sets have low standard Inception Scores. This is mainly because the Inception net is trained on ImageNet, which has a very different data distribution from the MNIST dataset. Based on this, we argue that the standard Inception Score is not suitable for some image datasets.

Fig. 5.6 shows the generation results for MNIST-TWO. Similarly, the model is also able to generate background and the two foreground objects separately. The foreground generator tends to generate a single digit at each timestep. Meanwhile, it captures the context information from the previous time steps. When the first digit is placed to the left side, the second one tends to be placed on the right side, and vice versa.

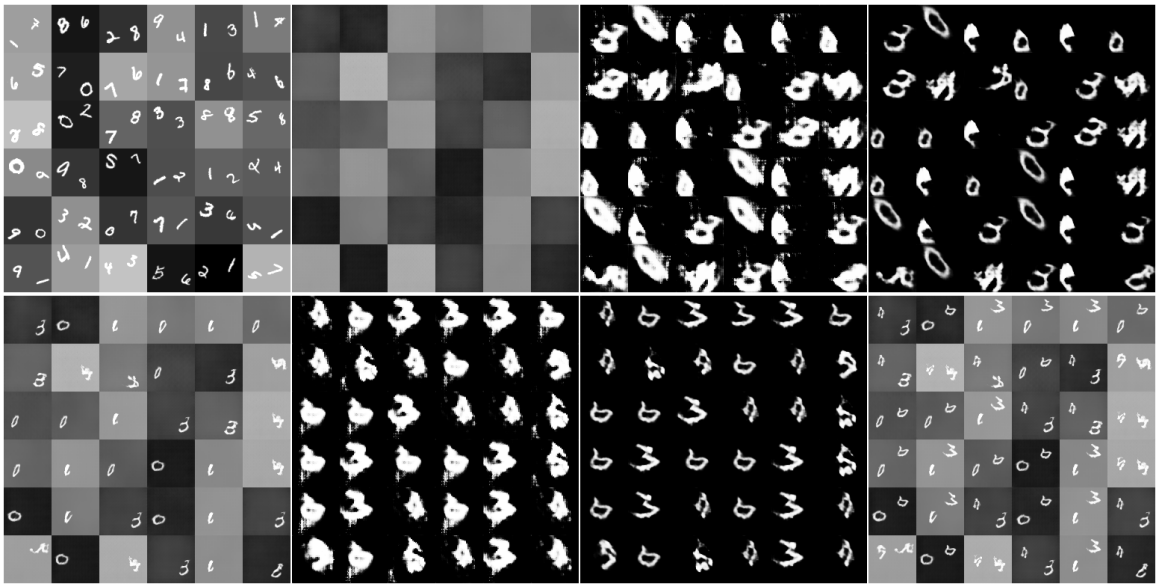


Figure 5.6: Generation results of our model on MNIST-TWO. From top left to bottom right (row major), the image blocks are real images, generated background images, foreground images and masks at the second timestep, composite images at the second time step, generated foreground images and masks at the third timestep and the final composite images, respectively.

CUB-200. We study the effectiveness of our model trained on the CUB-200 bird dataset. In Fig. 5.1, we have shown a random set of generated images, along with the intermediate generation results of the model. While being *completely unsupervised*, the model, for a large fraction of the samples, is able to successfully disentangle the foreground and the background. This is evident from the generated bird-like masks.

We do a comparative study based on Amazon Mechanical Turk (AMT) between DCGAN and LR-GAN to quantify relative visual quality of the generated images. We first generated 1000 samples from both the models. Then, we performed perfect matching between the two image sets using the Hungarian algorithm on $L2$ norm distance in the pixel space. This resulted in 1000 image pairs. Some examplar pairs are shown in Fig. 5.7. For each image pair, 9 judges are asked to choose the one that is more realistic. Based on majority voting, we find that our generated images are selected 68.4% times, compared with 31.6% times for DCGAN. This demonstrates that our model has generated more realistic images than DCGAN. We can attribute this difference to our model's ability to generate

Figure 5.7: Matched pairs of generated images based on DCGAN and LR-GAN, respectivedly. The odd columns are generated by DCGAN, and the even columns are generated by LR-GAN. These are paired according to the perfect matching based on Hungarian algorithm.

foreground separately from the background, enabling stronger edge cues.

In this experiment, we reduce the minimal allowed object scale to 1.1, which allows the model to generate larger foreground objects. The results are shown in Fig. 5.8. Similar to the results when the constraint is 1.2, the crisp bird-like masks are generated automatically by our model.

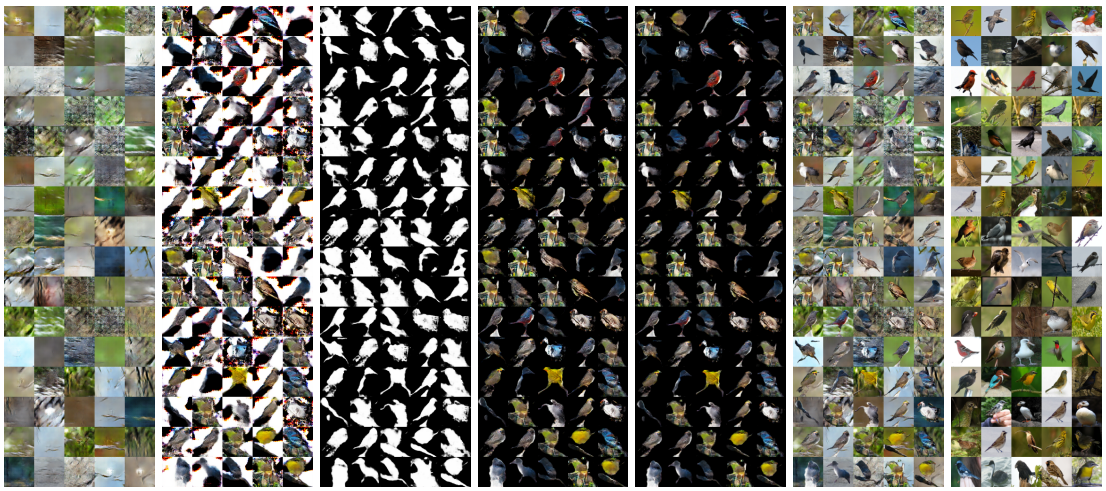


Figure 5.8: Generation results of our model on CUB-200 when setting minimal allowed scale to 1.1. From left to right, the blocks show the generated background images, foreground images, foreground masks, foreground images carved out by masks, carved foreground images after spatial transformation. The sixth and seventh blocks are final composite images and the nearest neighbor real images.

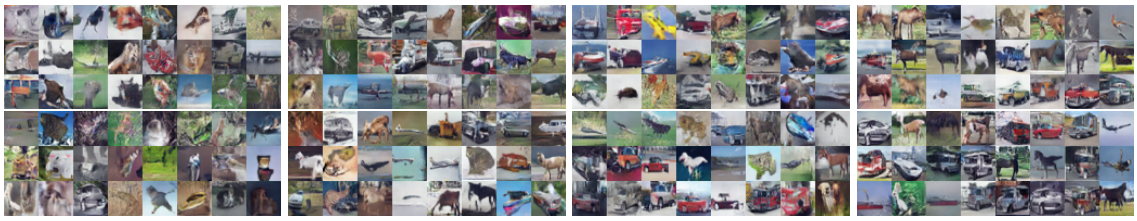


Figure 5.9: Qualitative comparison on CIFAR-10. Top three rows are images generated by DCGAN; Bottom three rows are by LR-GAN. From left to right, the blocks display generated images with increasing quality level as determined by human studies.

CIFAR-10. We now qualitatively and quantitatively evaluate our model on CIFAR-10, which contains multiple object categories and also various backgrounds.

Comparison of image generation quality: We conduct AMT studies to compare the fidelity of image generation. Towards this goal, we generate 1000 images from DCGAN and LR-GAN, respectively. We ask 5 judges to label each image to either belong to one of the 10 categories or as 'non recognizable' or 'recognizable but not belonging to the listed categories'. We then assign each image a quality level between [0,5] that captures the number of judges that agree with the majority choice. Fig. 5.9 shows the images generated by both approaches, ordered by increasing quality level. We merge images at quality level 0 (all judges said non-recognizable) and 1 together, and similarly images at level 4 and 5. Visually, the generated samples by our model have clearer boundaries and object structures. We also computed the fraction of non-recognizable images: Our model had a 10% absolute drop in non-recognizability rate (67.3% for ours vs. 77.7% for DCGAN). For reference, 11.4% of real CIFAR images were categorized as non-recognizable. Fig. 5.10

Table 5.3: Quantitative comparison between DCGAN and LR-GAN on CIFAR-10.

Training Data	Real Images	DCGAN	Ours
Inception Score†	11.18±0.18	6.64±0.14	7.17±0.07
Inception Score††	7.23±0.09	5.69±0.07	6.11±0.06
Adversarial Accuracy	83.33±0.08	37.81±0.02	44.22 ±0.08
Adversarial Divergence	0	7.58±0.04	5.57±0.06

†Evaluated using the pre-trained Inception net as [169]

††Evaluated using the supervisedly trained classifier based on the discriminator in LR-GAN.

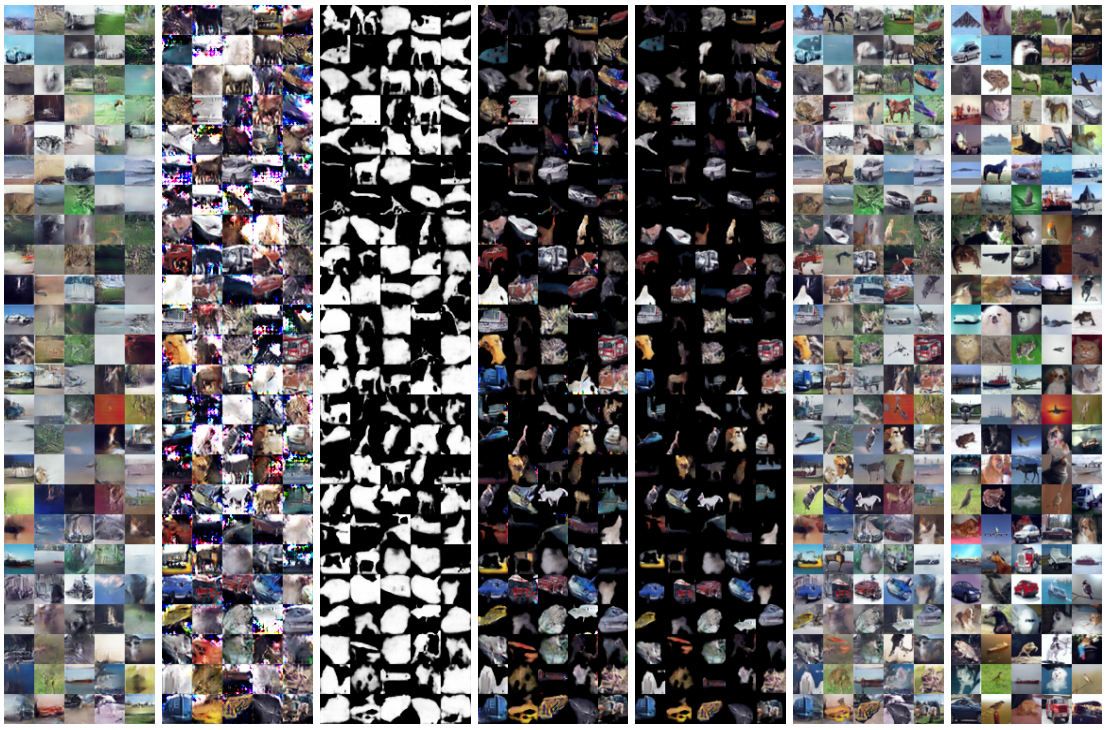


Figure 5.10: Generation results of our model on cifar-10. From left to right, the blocks are: generated background images, foreground images, foreground masks, foreground images carved out by masks, carved foregrounds after spatial transformation, final composite images and nearest neighbor training images to the generated images.

shows more generated (intermediate) results of our model. In Fig. 5.11, we show more results on CIFAR-10 when setting minimal allowed object scale to 1.1. The rightmost column block also shows the training images that are closest to the generated images (cosine similarity in pixel space). We can see our model does not memorize the training data.

Quantitative evaluation on generators: We evaluate the generators based on three metrics: 1) Inception Score; 2) Adversarial Accuracy; 3) Adversarial Divergence. blackTo obtain a classifier model for evaluation, we remove the top layer in the discriminator used in our model, and then append two fully connected layers on the top of it. We train this classifier using the training samples of CIFAR-10 based on the annotations. Following [169], we generated 50,000 images based on DCGAN and LR-GAN, repsectively. We compute two types of Inception Scores. The standard Inception Score is based on the Inception net as in [169], and the contextual Inception Score is based on our trained classifier model. To

Figure 5.11: Generation results of our model on cifar-10 with minimal allowed scale be 1.1, From left to right, the layout is same to Fig. 5.8.

distinguish, we denote the standard one as 'Inception Score†', and the contextual one as 'Inception Score††'. To obtain the Adversarial Accuracy and Adversarial Divergence scores, we train one generator on each of 10 categories for DCGAN and LR-GAN, respectively. Then, we use these generators to generate samples of different categories. Given these generated samples, we train the classifiers for DCGAN and LR-GAN separately. Along with the classifier trained on the real samples, we compute the Adversarial Accuracy and Adversarial Divergence on the real training samples. In Table 5.3, we report the Inception Scores, Adversarial Accuracy and Adversarial Divergence for comparison. We can see that our model outperforms DCGAN across the board. To point out, we obtan different Inception Scores based on different classifier models, which indicates that the Inception Score varies with different models.

Quantitative evaluation on discriminators: We evaluate the discriminator as an extractor for deep representations. Specifically, we use the output of the last convolutional

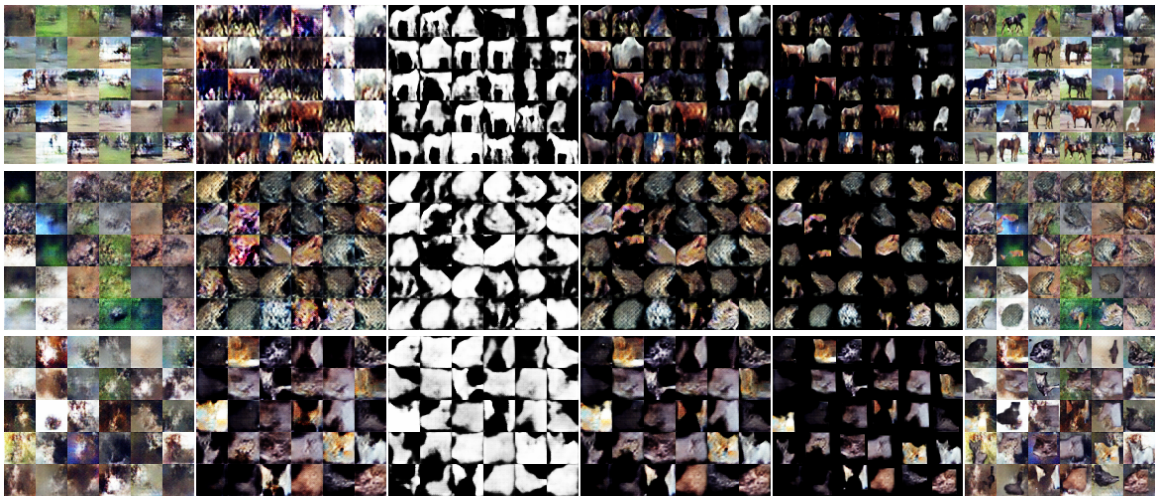


Figure 5.12: Category specific generation results of our model on CIFAR-10 categories of horse, frog, and cat (top to bottom). The blocks from left to right are: generated background images, foreground images, foreground masks, foreground images carved out by masks, carved foregrounds after spatial transformation and final composite images.

layer in the discriminator as features. We perform a 1-NN classification on the test set given the full training set. Cosine similarity is used as the metric. On the test set, our model achieves 62.09%$\pm$0.01% compared to DCGAN's 56.05%$\pm$0.02%.

Category specific models: The objects in CIFAR-10 exhibit huge variability in shapes. That can partly explain why some of the generated shapes are not as compelling in Fig. 5.10. To test this hypothesis, we reuse the generators trained for each of 10 categories used in our metrics to obtain the generation results. Fig. 5.12 shows results for categories 'horse', 'frog' and 'cat'. We can see that the model is now able to generate object-specific appearances and shapes, similar in vein to our results on the CUB-200 dataset.

Contextual generation: We also show the efficacy of our approach to generate diverse foregrounds conditioned on fixed background. The results in Fig. 5.13 in Appendix showcase that the foreground generator generates objects that are compatible with the background. This indicates that the model has captured contextual dependencies between the image layers.

Walking in the latent space: Similar to DCGAN, we also show results by walking in the latent space. Note that our model has two or more inputs. So we can walk along

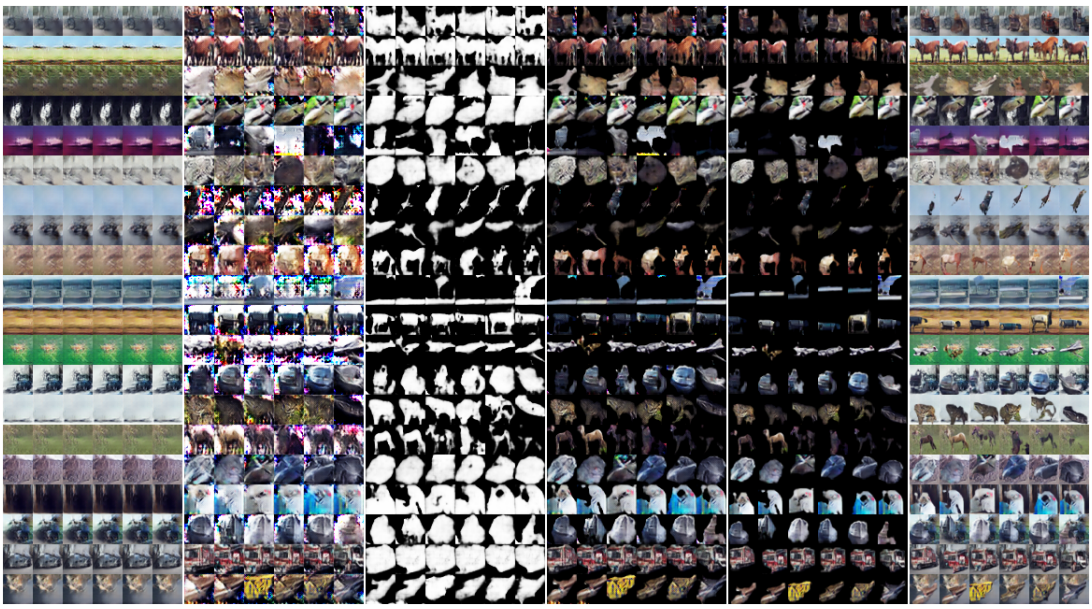


Figure 5.13: Walking in the latent foreground space by fixing backgrounds in our model on cifar-10. From left to right, the blocks are: generated background images, foreground images, foreground masks, foreground images carved out by masks, carved out foreground images after spatial transformation, and final composite images. Each row has the same background, but different foregrounds.

any of them or their combination. In Fig. 5.13, we generate multiple foregrounds for the same fixed generated background. We find that our model consistently generates contextually compatible foregrounds. For example, for the grass-like backgrounds, the foreground generator generates horses and deer, and airplane-like objects for the blue sky.

Figure 5.14: Statistics of annotations in human studies on cifar-10. Left to right: word cloud for real images, images generated by DCGAN, images generated by LR-GAN.

Word cloud based on human study: As we mentioned above, we conducted human studies on cifar-10. Besides asking people to select a name from a list for an image, we also conducted another human study where we ask people to use one word (free-form)

to describe the main object in the image. Each image was 'named' by 5 unique people. We generate word clouds for real images, images generated by DCGAN and LR-GAN, as shown in Fig. 5.14.

5.5.2 Importance of Transformations

Figure 5.15: Generation results from an ablated LR-GAN model without affine transformations. From top to bottom, the block rows correspond to different datasets: MNIST-ONE, CUB-200, CIFAR-10. From left to right, the blocks show generated background images, foreground images, foreground masks, and final composite images. For comparison, the rightmost column block shows final generated images from a non-ablated model with affine transformations.

Fig. 5.15 shows results from an ablated model without affine transformations in the foreground layers, and compares the results with the full model that does include these transformations. We note that one significant problem emerges that the decompositions are degenerate, in the sense that the model is unable to break the symmetry between foreground and background layers, often generating object appearances in the model's background layer and vice versa. For CUB-200, the final generated images have some blendings between foregrounds and backgrounds. This is particularly the case for those images with-

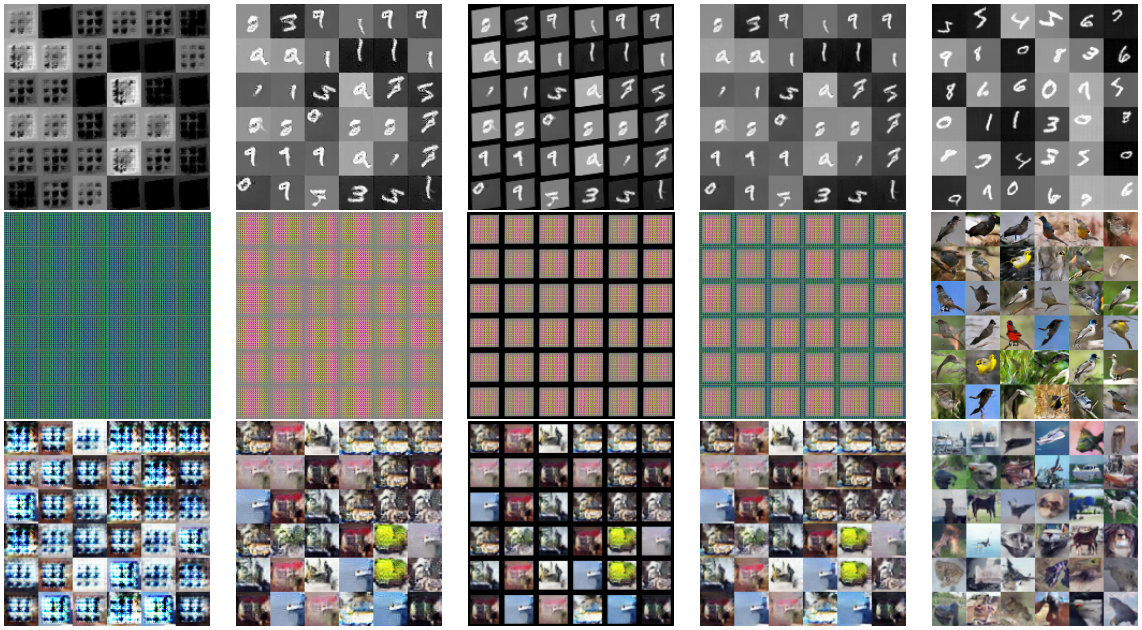


Figure 5.16: Generation results from an ablated LR-GAN model without mask generator. The block rows correspond to different datasets (from top to bottom: MNIST-ONE, CUB-200, CIFAR-10). From left to right, the blocks show generated background images, foreground images, transformed foreground images, and final composite images. For comparison, the rightmost column block shows final generated images from a non-ablated model with mask generator.

out bird-shape masks. For CIFAR-10, a number of generated masks are inverted. In this case, the background images are carved out as the foreground objects. The foreground generator takes almost all the duty to generate the final images, which make it harder to generate images as clear as the model with transformation. From these comparisons, we qualitatively demonstrate the importance of modeling transformations in the foreground generation process. Another merit of using transformation is that the intermediate outputs of the model are more interpretable and faciliate to the downstreaming tasks, such as scene paring, which is demonstrated in Section 5.5.6.

5.5.3 Importance of Shapes

We perform another ablation study by removing the mask generator to understand the importance of modeling object shapes. In this case, the generated foreground is simply pasted on top of the generated background after being transformed. There is no alpha blend-

ing between the foregrounds and backgrounds. The generation results for three datasets, MNIST-ONE, CUB-200, CIFAR-10 are shown in Fig. 5.16. As we can see, though the model works well for the generation of MNIST-ONE, it fails to generate reasonable images across the other datasets. Particularly, the training does not even converge for CUB-200. Based on these results, we qualitatively demonstrate that mask generator in our model is fairly important to obtain plausible results, especially for realistic images.

5.5.4 Results on LFW face dataset

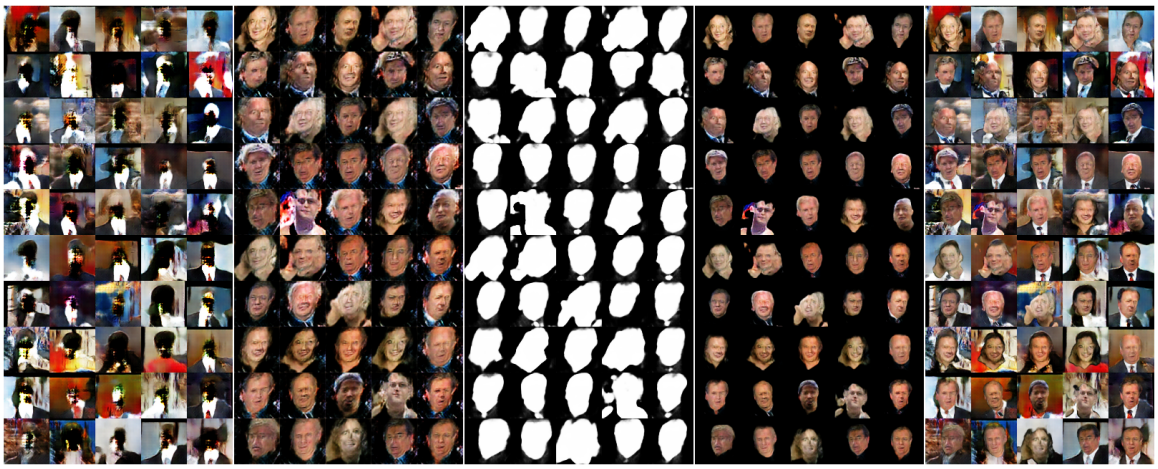


Figure 5.17: Generation results of our model on LFW. From left to right, the blocks are: generated background images, foreground images, foreground masks, carved out foreground images after spatial transformation, and final composite images.

We conduct experiment on face images in LFW dataset [195]. Different from previous works which work on cropped and aligned faces, we directly generate the original images which contains a large portion of backgrounds. This configuration helps to verify the efficiency of LR-GAN to model the object appearance, shape and pose. In Fig. 5.17, we show the (intermediate) generation results of LR-GAN. Surprisingly, without any supervisions, the model generated background and faces in separate steps, and the generated masks accurately depict face shapes. Moreover, the model learns where to place the generated faces so that the whole image looks natural. For comparison, please refer to [174] which does not model the transformation. We can find the generation results degrade much.

5.5.5 Statistics on Transformation Matrices

In this part, we analyze the statistics on the transformation matrices generated by our model for different datasets, including MNIST-ONE, CUB-200, CIFAR-10 and LFW. We used affine transformation in our model. So there are 6 parameters, scaling in the x coordinate (s_x), scaling in the y coordinate (s_y), translation in the x coordinate (t_x), translation in the y coordinate (t_y), rotation in the x coordinate (r_x) and rotation in the y coordinate (r_y). In Fig. 5.18, we show the histograms on different parameters for different datasets.These histograms show that the model produces non-trivial varied scaling, translation and rotation on all datasets. For different datasets, the learned transformation have different patterns. We hypothesize that this is mainly determined by the configurations of objects in the images. For example, on MNIST-ONE, all six parameters have some fluctuations since the synthetic dataset contains digits randomly placed at different locations. For the other three datasets, the scalings converge to single value since the object sizes do not vary much, and the variations on rotation and translation suffice to generate realistic images. Specifically, we can find the generator largely relies on the translation on x coordinate for generating CUB-200. This makes sense since birds in the images have similar scales, orientations but various horizontal locations. For CIFAR-10, since there are 10 different object categories, the configurations are more diverse, hence the generator uses all parameters for generation except for the scaling. For LFW, since faces have similar configurations, the learned transformations have less fluctuation as well. As a result, we can see that LR-GAN indeed models the transformations on the foreground to generate images.

5.5.6 Conditional Image Generation

Considering our model can generate object-like masks (shapes) for images, we conducted an experiment to evaluate whether our model can be potentially used for image segmentation and object detection. We make some changes to the model. For the background generator, the input is a real image instead of a random vector. Then the image is passed

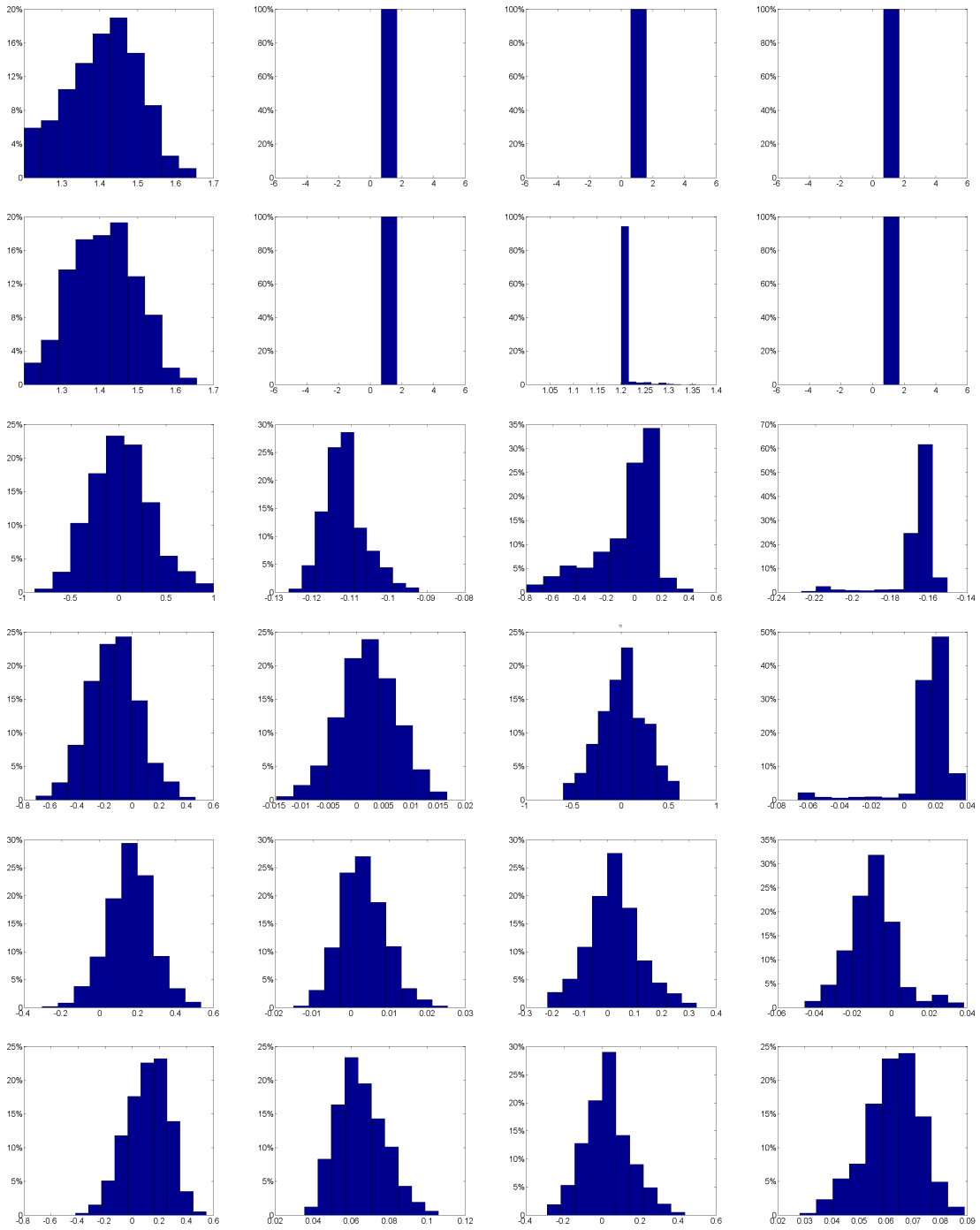


Figure 5.18: Histograms of transformation parameters learnt in our model for different datasets. From left to right, the datasets are: MNIST-ONE, CUB-200, CIFAR-10 and LFW. From top to bottom, they are scaling s_x, s_y, translation t_x, t_y, and rotation r_x, r_y in x and y coordinate, respectively.

through an encoder to extract the hidden features, which replaces the random vector z_0 and are fed to the background generator. For the foreground generator, we subtract the image generated by the background generator from the input image to obtain a residual image. Then this residual image is fed to the same encoder to get the hidden features, which are used as the input for foreground generator. In our conditional model, we want to reconstruct the image, so we add a reconstruction loss along with the adversarial loss. We train this conditional model on CIFAR-10. The (intermediate) outputs of the model is shown in Fig. 5.19. Interestingly, the model successfully learned to decompose the input images into background and foreground. The background generator tends to do an image inpainting by generating a complete background without object, while the foreground generator works as a segmentation model to get object mask from the input image.

Similarly, we also run the conditional LR-GAN on LFW dataset. As we can see in Fig. 5.19, the foreground generator automatically and consistently learned to generate the face regions, even though there are large portion of background in the input images. In other words, the conditional LR-GAN successfully learned to detection faces in images. We suspect this success is due to that it has low cost for the generator to generate similar images, and thus converge to the case that the first generator generate background, and the second generator generate face images.

5.6 Discussion

In this chapter, we proposed a layered recursive image generation method. Unlike previous work, we explicitly exploit the structure in images and disentangle the process into background and foreground generation separately. The generated background and foregrounds are then combined to the final image by considering the spatial relationships. We also proposed two new evaluation metrics, adversarial divergence and adversarial accuracy to evaluate the quality of generated images. In the experiments, we demonstrated that our generated images have higher quality in terms of automatic metrics and human studies.

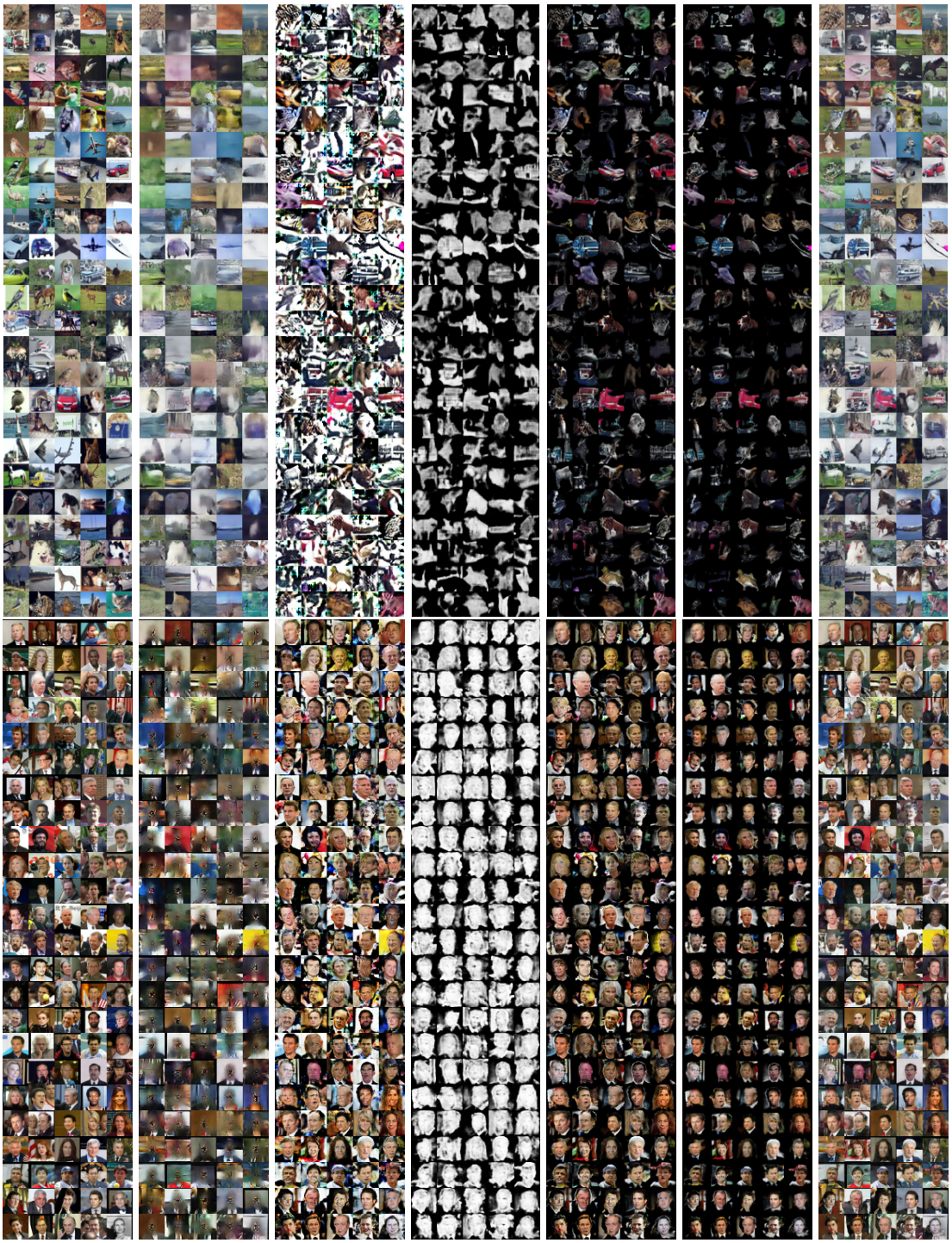


Figure 5.19: Conditional generation results on cifar-10 and LFW. From left to right, the blocks are: real images, generated background images, foreground images, foreground masks, foreground images carved out by masks, carved foreground images after spatial transformation, and final composite (reconstructed) images.

Part III

Structured Visual Reasoning

CHAPTER 6

REASON ON OBJECT ENTITIES FOR IMAGE CAPTIONING

In this chapter, I will discuss about how we make use of the image-level structure for image captioning. As a description of image content, image caption is aimed at describing the image with a compact but comprehensive sentence. As such, an image caption usually cover a number of object entities in the image. Recently, most of image captioning models still reply on the holistic image representation to generate an image captioning. In this work, we resort to a more fine-grained and structured representation of an image and perform reasoning on top of this structured representation for image captioning. Specifically, we extract the object entities from a given image, and then use a slot-fitting method to determine whether and which object entity should be inserted in the blank of a sentence. This way, we disentangle the image captioning into two separate parts, one for structured visual understanding and one for reasoning for generating the caption. Our experiments show that the proposed image captioning model not only outperforms previous work but also achieve a good generalization ability to novel scenarios.

6.1 Introduction

Image captioning is a challenging problem that lies at the intersection of computer vision and natural language processing. It involves generating a natural language sentence that accurately summarizes the contents of an image. Image captioning is also an important first step towards real-world applications with significant practical impact, ranging from aiding visually impaired users to personal assistants to human-robot interaction [16, 196].

State-of-art image captioning models today tend to be monolithic neural models, essentially of the "encoder-decoder" paradigm. Images are encoded into a vector with a convolutional neural network (CNN), and captions are decoded from this vector using a

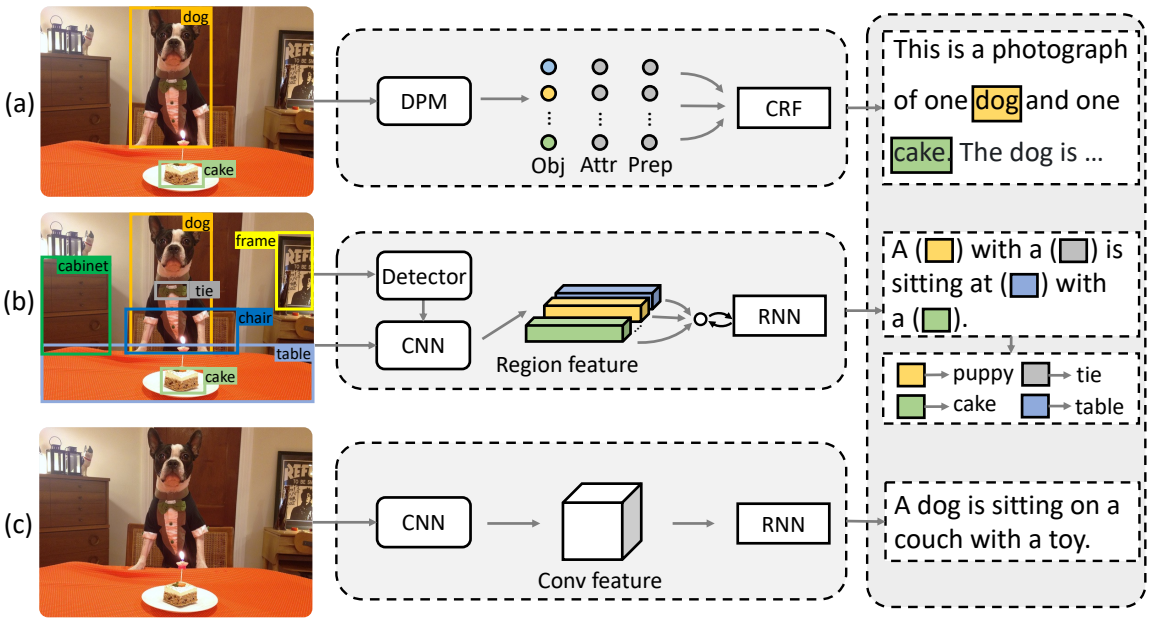


Figure 6.1: Example captions generated by (a) Baby Talk [202], (c) neural image captioning [203] and (b) our Neural Baby Talk approach. Our method generates the sentence "template" with slot locations (illustrated with filled boxes) explicitly tied to image regions (drawn in the image in corresponding colors). These slots are then filled by object detectors with concepts found in regions.

Recurrent Neural Network (RNN), with the entire system trained end-to-end. While there are many recent extensions of this basic idea to include attention [34, 197, 198, 199, 200], it is well-understood that models still lack visual grounding (*i.e.*, do not associate named concepts to pixels in the image). They often tend to 'look' at different regions than humans would and tend to copy captions from training data [201].

For instance, in Fig. 6.1 a neural image captioning approach [203] describes the image as "A dog is sitting on a couch with a toy." This is not quite accurate. But if one were to *really* squint at the image, it (arguably) does perhaps look like a scene where a dog *could* be sitting on a couch with a toy. It certainly is common to find dogs sitting on couches with toys. A-priori, the description is reasonable. That's exactly what today's neural captioning models tend to do – produce generic *plausible* captions based on the language model[1] that match a first-glance gist of the scene. While this may suffice for common scenes, images

[1] frequently, directly reproduced from a caption in the training data.

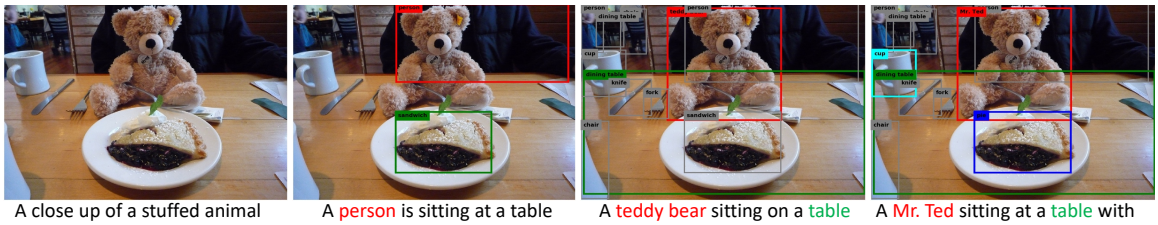


Figure 6.2: From left to right is the generated caption using the same captioning model but with different detectors: 1) No detector; 2) A weak detector that only detects "person" and "sandwich"; 3) A detector trained on COCO [71] categories (including "teddy bear"). 4) A detector that can detect novel concepts (e.g. "Mr. Ted" and "pie" that never occurred in the captioning training data). Different colors show a correspondence between the visual word and grounding regions.

that differ from canonical scenes – given the diversity in our visual world, there are *plenty* of such images – tend to be underserved by these models.

If we take a step back – do we really need the language model to do the heavy lifting in image captioning? Given the unprecedented progress we are seeing in object recognition[2] (e.g., object detection, semantic segmentation, instance segmentation, pose estimation), it seems like the vision pipeline can certainly do better than rely on just a first-glance gist of the scene. In fact, today's state-of-the-art object detectors can successfully detect the table and cake in the image in Fig. 6.1(c)! The caption ought to be able to talk about the table and cake *actually detected* as opposed to letting the language model hallucinate a couch and a toy simply because that sounds plausible.

Interestingly, some of the first attempts at image captioning [204, 202, 205, 206] – before the deep learning "revolution" – relied heavily on outputs of object detectors and attribute classifiers to describe images. For instance, consider the output of Baby Talk [202] in Fig. 6.1, that used a slot filling approach to talk about all the objects and attributes found in the scene via a templated caption. The language is unnatural but the caption is very much grounded in what the model sees in the image. Today's approaches fall at the other extreme on the spectrum – the language generated by modern neural image captioning approaches

[2]e.g., 11% absolute increase in average precision in object detection in the COCO challenge in the last year.

is much more natural but tends to be much less grounded in the image.

In this paper, we introduce Neural Baby Talk that reconciles these methodologies. It produces natural language *explicitly* grounded in entities found by object detectors. It is a neural approach that generates a sentence "template" with slot locations explicitly tied to image regions. These slots are then filled by object recognizers with concepts found in the regions. The entire approach is trained end-to-end. This results in natural sounding and grounded captions.

Our main technical contribution is a novel neural decoder for grounded image captioning. Specifically, at each time step, the model decides whether to generate a word from the textual vocabulary or generate a "visual" word. The visual word is essentially a token that will hold the slot for a word that is to describe a specific region in the image. For instance, for the image in Fig. 6.1, the generated sequence may be "A <region−17> is sitting at a <region−123> with a <region−3>." The visual words (<region−[.]>'s) are then filled in during a second stage that classifies each of the indicated regions (e.g., <region−17>→puppy, <region−123>→table), resulting in a final description of "A puppy is sitting at a table with a cake." – a free-form natural language description that is grounded in the image. One nice feature of our model is that it allows for different object detectors to be plugged in easily. As a result, a variety of captions can be produced for the same image using different detection backends. See Fig. 6.2 for an illustration.

Contributions: Our contributions are as follows:

- We present Neural Baby Talk – a novel framework for visually grounded image captioning that explicitly localizes objects in the image while generating free-form natural language descriptions.
- Ours is a two-stage approach that first generates a hybrid template that contains a mix of (text) words and slots explicitly associated with image regions, and then fills in the slots with (text) words by recognizing the content in the corresponding image regions.

- We propose a robust image captioning task to benchmark compositionality of image captioning algorithms where at test time the model encounters images containing known objects but in novel combinations (e.g., the model has seen dogs on couches and people at tables during training, but at test time encounters a dog at a table). Generalizing to such novel compositions is one way to demonstrate image grounding as opposed to simply leveraging correlations from training data.
- Our proposed method achieves state-of-the-art performance on COCO and Flickr30k datasets on the standard image captioning task, and significantly outperforms existing approaches on the robust image captioning and novel object captioning tasks.

6.2 Background

Some of the earlier approaches generated templated image captions via slot-filling. For instance, Kulkarni *et al*. [202] detect objects, attributes, and prepositions, jointly reason about these through a CRF, and finally fill appropriate slots in a template. Farhadi *et al*.[204] compute a triplet for a scene, and use this templated "meaning" representation to retrieve a caption from a database. [205, 206] use more powerful language templates such as a syntactically well-formed tree. These approaches tend to either produce captions that are relevant to the image but not natural sounding, or captions that are natural (*e.g*.retrieved from a database of captions) but may not be sufficiently grounded in the image.

Neural models for image captioning have been receiving increased attention in the last few years [207, 208, 209, 210, 211, 203]. State-of-the-art neural approaches include attention mechanisms [34, 197, 198, 199, 200, 212, 37] that identify regions in the image to "ground" emitted words. In practice, these attention regions tend to be quite blurry, and rarely correspond to semantically meaningful individual entities (e.g., objects instances) in the image. Our approach grounds words in object detections, which by design identify concrete semantic entities (object instances) in the image.

There has been some recent interest in grounding natural language in images. Dense

Captioning [213] generates descriptions for specific image regions. In contrast, our model produces captions for the entire image, with words grounded in concrete entities in the image. Another related line of work is on resolving referring expressions [214] (or description-based object retrieval [215, 216, 217, 218] – given a description of an object in the image, identify which object is being referred to) or referring expression generation [214, 219, 220, 221] (given an object in the image, generate a discriminative description of the object). While the interest in grounded language is in common, our task is different.

One natural strength of our model is its ability to incorporate different object detectors, including the ability to generate captions with novel objects (never seen before in training captions). In that context, our work is related to prior works on novel object captioning [222, 223, 224, 225]. As we describe in Sec. 6.4.3, our method outperforms these approaches by 14.6% on the averaged F1 score.

6.3 Method

Given an image $\boldsymbol{I}$, the goal of our method is to generate visually grounded descriptions $\boldsymbol{y} = \{y_1, \ldots, y_T\}$. Let $\boldsymbol{r_I} = \{r_1, ..., r_N\}$ be the set of N images regions extracted from $\boldsymbol{I}$. When generating an entity word in the caption, we want to ground it in a specific image region $r \in \boldsymbol{r_I}$. Following the standard supervised learning paradigm, we learn parameters $\boldsymbol{\theta}$ of our model by maximizing the likelihood of the correct caption:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{(\boldsymbol{I},\boldsymbol{y})} \log p(\boldsymbol{y}|\boldsymbol{I};\boldsymbol{\theta}) \tag{6.1}$$

Using chain rule, the joint probability distribution can be decomposed over a sequence of tokens:

$$p(\boldsymbol{y}|\boldsymbol{I}) = \prod_{t=1}^{T} p(y_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{I}) \tag{6.2}$$

where we drop the dependency on model parameters to avoid notational clutter. We introduce a latent variable r_t to denote a specific image region so that y_t can explicitly

ground in it. Thus the probability of y_t is decomposed to:

$$p(y_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{I}) = p(y_t|r_t, \boldsymbol{y}_{1:t-1}, \boldsymbol{I})p(r_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{I}) \tag{6.3}$$

In our framework, y_t can be of one of two types: a visual word or a textual word, denoted as y^{vis} and y^{txt} respectively. A visual word y^{vis} is a type of word that is grounded in a specific image region drawn from $\boldsymbol{r_I}$. A textual word y^{txt} is a word from the remainder of the caption. It is drawn from the language model , which is associated with a "default" sentinel "region" $\tilde{r}$ obtained from the language model [200] (discussed in Sec. 6.3.1). For example, as illustrated in Fig. 7.1, "puppy" and "cake" grounded in the bounding box of category "dog" and "cake" respectively, are visual words. While "with" and "sitting" are not associated with any image regions and thus are textual words.

With this, Eq. 6.1 can be decomposed into two cascaded objectives. First, maximizing the probability of generating the sentence "template". A sequence of grounding regions associated with the visual words interspersed with the textual words can be viewed as a sentence "template", where the grounding regions are slots to be filled in with visual words.[3] An example template (Fig. 6.3) is "A <region−2> is laying on the <region−4> near a <region−7>. Second, maximizing the probability of visual words y_t^{vis} conditioned on the grounding regions and object detection information, e.g., categories recognized by detector. In the template example above, the model will fill the slots with 'cat', 'laptop' and 'chair' respectively.

In the following, we first describe how we generate the slotted caption template (Sec. 6.3.1), and then how the slots are filled in to obtain the final image description (Sec. 6.3.2). The overall objective function is described in Sec. 6.3.3 and the implementation details in Sec. 6.3.4.

[3]Our approach is not limited to any pre-specified bank of templates. Rather, our approach automatically generates a template (with placeholders – slots – for visually grounded words), which may be any one of the exponentially many possible templates.

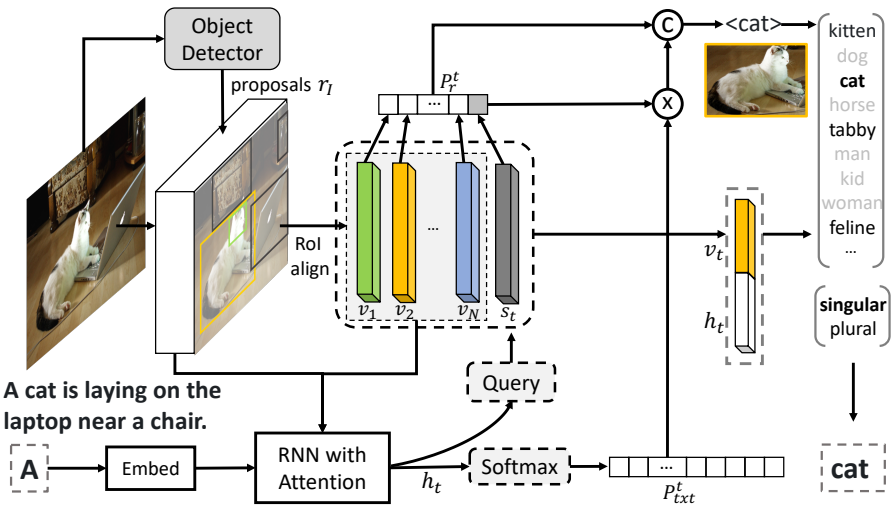


Figure 6.3: One block of the proposed approach. Given an image, proposals from any object detector and current word "A", the figure shows the process to predict the next visual word "cat".

6.3.1 "Slotted" Caption Template Generation

Given an image $\boldsymbol{I}$, and the corresponding caption $\boldsymbol{y}$, the candidate grounding regions are obtained by using a pre-trained Faster-RCNN network [51]. To generate the caption "template", we use a recurrent neural network, which is commonly used as the decoder for image captioning [208, 210]. At each time step, we compute the RNN hidden state $\boldsymbol{h}_t$ according to the previous hidden state $\boldsymbol{h}_{t-1}$ and the input $\boldsymbol{x}_t$ such that $\boldsymbol{h}_t = \mathrm{RNN}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$. At training time, x_t is the ground truth token (teacher forcing) and at test time is the sampled token y_{t-1}. Our decoder consists of an attention based LSTM layer [212] that takes convolution feature maps as input. Details can be found in Sec. 6.3.4. To generate the "slot" for visual words, we use a pointer network [226] that modulates a content-based attention mechanism over the grounding regions. Let $\boldsymbol{v_t} \in \mathcal{R}^{d\times 1}$ be the region feature of r_t, which is calculated based on Faster R-CNN. We compute the pointing vector with:

$$u_i^t = \boldsymbol{w}_h^T \tanh(\boldsymbol{W}_v \boldsymbol{v}_t + \boldsymbol{W}_z \boldsymbol{h}_t) \tag{6.4}$$

$$\boldsymbol{P}_{\boldsymbol{r_I}}^t = \mathrm{softmax}(\boldsymbol{u}^t) \tag{6.5}$$

where $\boldsymbol{W}_v \in \mathbb{R}^{m \times d}$, $\boldsymbol{W}_z \in \mathbb{R}^{d \times d}$ and $\boldsymbol{w}_h \in \mathbb{R}^{d \times 1}$ are parameters to be learned. The softmax normalizes the vector $\boldsymbol{u}^t$ to be a distribution over grounding regions $\boldsymbol{r_I}$.

Since textual words y_t^{txt} are not tied to specific regions in the image, inspired by [200], we add a "visual sentinel" $\tilde{r}$ as a latent variable to serve as dummy grounding for the textual word. The visual sentinel can be thought of as a latent representation of what the decoder already knows about the image. The probability of a textual word y_t^{txt} then is:

$$p(y_t^{txt}|\boldsymbol{y}_{1:t-1}) = p(y_t^{txt}|\tilde{r}, \boldsymbol{y}_{1:t-1})p(\tilde{r}|\boldsymbol{y}_{1:t-1}) \tag{6.6}$$

where we drop the dependency on $\boldsymbol{I}$ to avoid clutter.

We first describe how the visual sentinel is computed, and then how the textual words are determined based on the visual sentinel. Following [200], when the decoder RNN is an LSTM [227], the representation for visual sentinel $\boldsymbol{s}_t$ can be obtained by:

$$\boldsymbol{g}_t = \sigma\left(\boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{W}_h \boldsymbol{h}_{t-1}\right) \tag{6.7}$$

$$\boldsymbol{s}_t = \boldsymbol{g}_t \odot \tanh\left(\boldsymbol{c}_t\right) \tag{6.8}$$

where $\boldsymbol{W}_x \in \mathbb{R}^{d \times d}$, $\boldsymbol{W}_h \in \mathbb{R}^{d \times d}$. $\boldsymbol{x}_t$ is the LSTM input at time step t, and $\boldsymbol{g}_t$ is the gate applied on the cell state $\boldsymbol{c}_t$. $\odot$ represents element-wise product, σ the logistic sigmoid activation. Modifying Eq. 6.5, the probability over the grounding regions including the visual sentinel is:

$$\boldsymbol{P}_{\boldsymbol{r}}^t = \text{softmax}([\boldsymbol{u}^t; \boldsymbol{w}_h^T \tanh(\boldsymbol{W}_s \boldsymbol{s}_t + \boldsymbol{W}_z \boldsymbol{h}_t)]) \tag{6.9}$$

where $\boldsymbol{W}_s \in \mathbb{R}^{d \times d}$ and $\boldsymbol{W}_z \in \mathbb{R}^{d \times d}$ are the parameters. Notably, $\boldsymbol{W}_z$ and $\boldsymbol{w}_h$ are the same parameters as in Eq. 6.4. $\boldsymbol{P}_{\boldsymbol{r}}^t$ is the probability distribution over grounding regions $\boldsymbol{r_I}$ and visual sentinel $\tilde{r}$. The last element of the vector in Eq. 6.9 captures $p(\tilde{r}|\boldsymbol{y}_{1:t-1})$.

We feed the hidden state $\boldsymbol{h}_t$ into a softmax layer to obtain the probability over textual

words conditioned on the image, all previous words, and the visual sentinel:

$$\boldsymbol{P}_{txt}^{t} = \text{softmax}\left(\boldsymbol{W}_q \boldsymbol{h}_t\right) \tag{6.10}$$

where $\boldsymbol{W}_q \in \mathbb{R}^{V \times d}$, d is hidden state size, and V is textual vocabulary size. Plugging in Eq. 6.10 and $p(\tilde{r}|\boldsymbol{y}_{1:t-1})$ from the last element of the vector in Eq. 6.9 into Eq. 6.6 gives us the probability of generating a textual word in the template.

6.3.2 Caption Refinement: Filling in The Slots

To fill the slots in the generated template with visual words grounded in image regions, we leverage the outputs of an object detection network. Given a grounding region, the category can be obtained through any detection framework [51]. But outputs of detection networks are typically singular coarse labels *e.g.*"dog". Captions often refer to these entities in a fine-grained fashion *e.g.*"puppy" or in the plural form "dogs". In order to accommodate for these linguistic variations, the visual word y^{vis} in our model is a refinement of the category name by considering the following two factors: First, determine the plurality – whether it should be singular or plural. Second, determine the fine-grained class (if any). Using two single layer MLPs with ReLU activation $f(\cdot)$, we compute them with:

$$\boldsymbol{P}_{b}^{t} = \text{softmax}\left(\boldsymbol{W}_b f_b\left([\boldsymbol{v}_t; \boldsymbol{h}_t]\right)\right) \tag{6.11}$$

$$\boldsymbol{P}_{g}^{t} = \text{softmax}\left(\boldsymbol{U}^T \boldsymbol{W}_g f_g\left([\boldsymbol{v}_t; \boldsymbol{h}_t]\right)\right) \tag{6.12}$$

$\boldsymbol{W}_b \in \mathbb{R}^{2 \times d}$, $\boldsymbol{W}_g \in \mathbb{R}^{300 \times d}$ are the weight parameters. $\boldsymbol{U} \in \mathbb{R}^{300 \times k}$ is the glove vector embeddings [**pennington2014glove**] for k fine-grained words associated with the category name. The visual word y_t^{vis} is then determined by plurality and fine-grained class (*e.g.*, if plurality is plural, and the fine-grained class is "puppy", the visual word will be "puppies").

6.3.3 Objective

Most standard image captioning datasets (*e.g.*COCO [71]) do not contain phrase grounding annotations, while some datasets do (*e.g.*Flickr30k [215]). Our training objective (presented next) can incorporate different kinds of supervision – be it strong annotations indicating which words in the caption are grounded in which boxes in the image, or weak supervision where objects are annotated in the image but are not aligned to words in the caption. Given the target ground truth caption $\boldsymbol{y}^*_{1:T}$ and a image captioning model with parameters $\boldsymbol{\theta}$, we minimize the cross entropy loss:

$$L(\boldsymbol{\theta}) = -\sum_{t=1}^{T} \log \Big(\overbrace{p(y^*_t|\tilde{r}, \boldsymbol{y}^*_{1:t-1}) p(\tilde{r}|\boldsymbol{y}^*_{1:t-1}) 1_{(y^*_t = y^{\text{txt}})}}^{\text{Textual word probability}} + \underbrace{p\left(b^*_t, s^*_t|\boldsymbol{r}_t, \boldsymbol{y}^*_{1:t-1}\right)}_{\text{Caption refinement}} \Big(\underbrace{\frac{1}{m}\sum_{i=1}^{m} p\left(r^i_t|\boldsymbol{y}^*_{1:t-1}\right) 1_{(y^*_t = y^{\text{vis}})}}_{\text{Averaged target region probability}} \Big) \Big) \tag{6.13}$$

where y^*_t is the word from the ground truth caption at time t. $1_{(y^*_t = y^{txt})}$ is the indicator function which equals to 1 if y^*_t is textual word and 0 otherwise. b^*_t and s^*_t are the target ground truth plurality and find-grained class. $\{r^i_t\}^m_{i=1} \in \boldsymbol{r_I}$ are the target grounding regions of the visual word at time t. We maximize the averaged log probability of the target grounding regions.

Visual word extraction. During training, visual words in a caption are dynamically identified by matching the base form of each word (using the Stanford lemmatization toolbox [**manning-EtAl:2014:P14-5**]) against a vocabulary of visual words (details of how to get visual word can be found in dataset Sec. 6.4). The grounding regions $\{r^i_t\}^m_{i=1}$ for a visual word y_t is identified by computing the IoU of all boxes detected by the object detection network with the ground truth bounding box associated with the category corresponding to y_t. If the score exceeds a threshold of 0.5 and the grounding region label matches the visual word, the bounding boxes are selected as the grounding regions. E.g., given a target visual

word “cat”, if there are no proposals that match the target bounding box, the model predicts the textual word “cat” instead.

6.3.4 Implementation Details

Detection model. We use Faster R-CNN [51] with ResNet-101 [9] to obtain region proposals for the image. We use an IoU threshold of 0.7 for region proposal suppression and 0.3 for class suppressions. A class detection confidence threshold of 0.5 is used to select regions.

Region feature. We use a pre-trained ResNet-101 [**he2015deep**] in our model. The image is first resized to 576×576 and we random crop 512×512 as the input to the CNN network. Given proposals from the pre-trained detection model, the feature $\boldsymbol{v}_i$ for region i is a concatenation of 3 different features $\boldsymbol{v}_i = [\boldsymbol{v}_i^p; \boldsymbol{v}_i^l; \boldsymbol{v}_i^g]$ where $\boldsymbol{v}_i^p$ is the pooling feature of RoI align layer [52] given the proposal coordinates, $\boldsymbol{v}_i^l$ is the location feature and $\boldsymbol{v}_i^g$ is the glove vector embedding of the class label for region i. Let $x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}$ be the bounding box coordinates of the region b; W_I and H_I be the width and height of the image I. Then the location feature $\boldsymbol{v}_i^l$ can be obtained by projecting the normalized location $[\frac{x_{\text{min}}}{W_I}, \frac{y_{\text{min}}}{H_I}, \frac{x_{\text{max}}}{W_I}, \frac{y_{\text{max}}}{H_I}]$ into another embedding space.

Language model. We use an attention model with two LSTM layers [37] as our base attention model. Given N region features from detection proposals $\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N\}$ and CNN features from the last convolution layer at K grids $\hat{\boldsymbol{V}} = \{\hat{\boldsymbol{v}}_1, \ldots, \hat{\boldsymbol{v}}_K\}$, the language model has two separate attention layers shown in Fig 6.4. The attention distribution over the image features for detection proposals is:

$$\begin{aligned} \boldsymbol{z}_t &= \boldsymbol{w}_z^T \tanh\left(\boldsymbol{W}_v \boldsymbol{V} + (\boldsymbol{W}_{\boldsymbol{g}} \boldsymbol{h}_{\boldsymbol{t}}) 1^T\right) \\ \boldsymbol{\alpha}_t &= softmax(\boldsymbol{z}_t) \end{aligned} \tag{6.14}$$

where $\boldsymbol{W}_v \in \mathbb{R}^{m \times d}$, $\boldsymbol{W}_g \in \mathbb{R}^{d \times d}$ and $\boldsymbol{w} \in \mathbb{R}^{d \times 1}$. $1 \in \mathbb{R}^N$ is a vector with all elements set to 1. $\boldsymbol{\alpha}_t$ is the attention weight over N image location features.

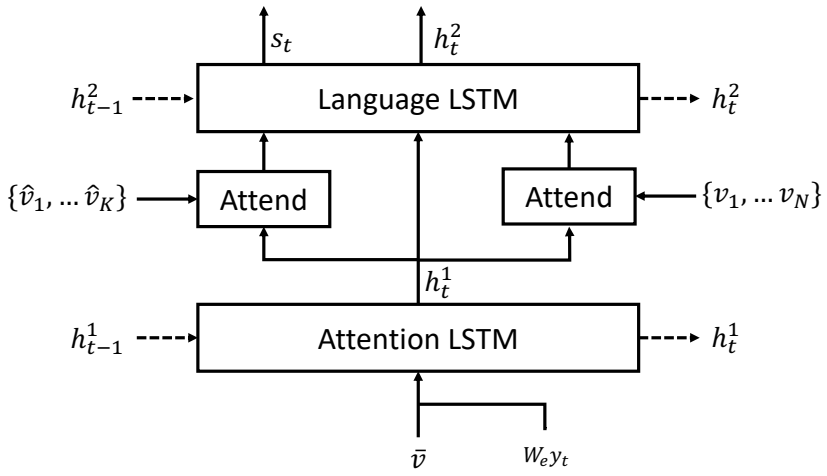


Figure 6.4: Language model used in our approach.

Training details. In our experiments, we use a two layer LSTM with hidden size 1024. The number of hidden units in the attention layer and the size of the input word embedding are 512. We use the Adam [228] optimizer with an initial learning rate of 5×10^{-4} and anneal the learning rate by a factor of 0.8 every three epochs. We train the model up to 50 epochs with early stopping. Note that we do not finetune the CNN network during training. We set the batch size to be 100 for COCO [71] and 50 for Flickr30k [215].

6.4 Experimental Results

Datasets. We experiment with two datasets. Flickr30k Entities [215] contains 275,755 bounding boxes from 31,783 images associated with natural language phrases. Each image is annotated with 5 crowdsourced captions. For each annotated phrase in the caption, we identify visual words by selecting the inner most NP (noun phrase) tag from the Stanford part-of-speech tagger [229].We use Stanford Lemmatization Toolbox [230] to get the base form of the entity words resulting in 2,567 unique words.

COCO [71] contains 82,783, 40,504 and 40,775 images for training, validation and testing respectively. Each image has around 5 crowdsourced captions. Unlike Flickr30k Entities, COCO does not have bounding box annotations associated with specific phrases

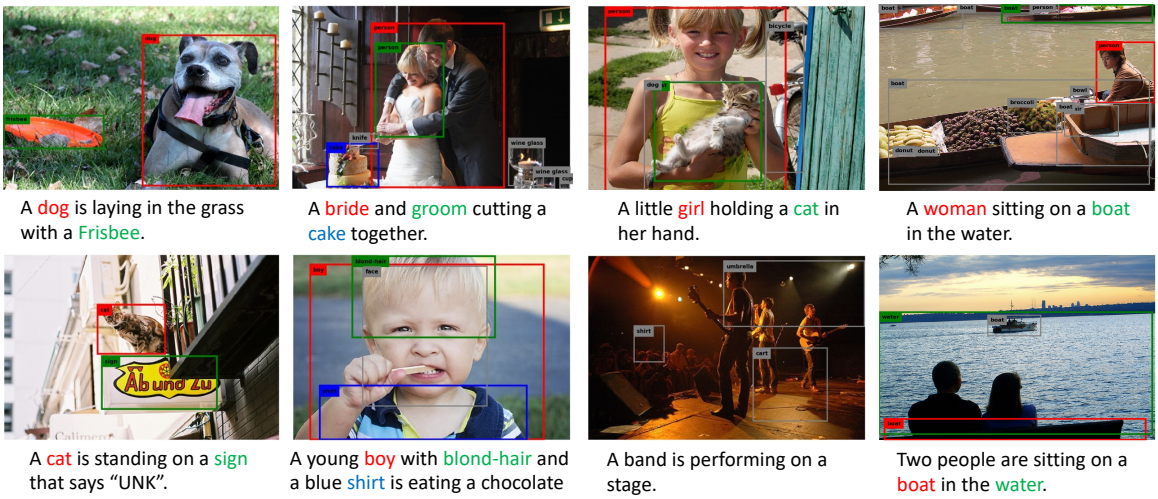


Figure 6.5: Generated captions and corresponding visual grounding regions on the standard image captioning task (Top: COCO, Bottom: Flickr30k). Different colors show a correspondence between the visual words and grounding regions. Grey regions are the proposals not selected in the caption. First 3 columns show success and last column shows failure cases (words are grounded in the wrong region).

Method	BLEU1	BLEU4	METEOR	CIDEr	SPICE
Hard-Attention [**Xu2015show**]	66.9	19.9	18.5	-	-
ATT-FCN [198]	64.7	23.0	18.9	-	-
Adaptive [200]	67.7	25.1	20.4	53.1	14.5
NBT	**69.0**	**27.1**	**21.7**	**57.5**	**15.6**
NBT$^{\text{oracle}}$	72.0	28.5	23.1	64.8	19.6

Table 6.1: Performance on the test portion of Karpathy *et al.* [203]'s splits on Flickr30k Entities dataset.

or entities in the caption. To identify visual words, we manually constructed an object category to word mapping that maps object categories like <person> to a list of potential fine-grained labels like ["child", "baker", ...]. This results in 80 categories with a total of 413 fine-grained classes. See supp. for details.

Detector pre-training. We use open an source implementation [73] of Faster-RCNN [51] to train the detector. For Flickr30K Entities, we use visual words that occur at least 100 times as detection labels, resulting in a total of 460 detection labels. Since detection labels and visual words have a one-to-one mapping, we do not have fine-grained classes for the Flickr30K Entities dataset – the caption refinement process only determines the plurality of

Method	BLEU1	BLEU4	METEOR	CIDEr	SPICE
Adaptive [200]	74.2	32.5	26.6	**108.5**	19.5
Att2in [212]	-	31.3	26.0	101.3	-
Up-Down [37]	74.5	33.4	26.1	105.4	19.2
Att2in* [212]	-	33.3	26.3	111.4	-
Up-Down† [37]	79.8	36.3	27.7	120.1	21.4
NBT	**75.5**	**34.7**	**27.1**	107.2	**20.1**
NBT$^{\text{oracle}}$	75.9	34.9	27.4	108.9	20.4

Table 6.2: Performance on the test portion of Karpathy *et al*. [203]'s splits on COCO dataset. $*$ directly optimizes the CIDEr Metric, † uses better image features, and are thus not directly comparable.

detection labels. For COCO, ground truth detection annotations are used to train the object detector.

Caption pre-processing. We truncate captions longer than 16 words for both COCO and Flickr30k Entities dataset. We then build a vocabulary of words that occur at least 5 times in the training set, resulting in 9,587 and 6,864 words for COCO and Flickr30k Entities, respectively.

6.4.1 Standard Image Captioning

For standard image captioning, we use splits from Karpathy *et al*. [203] on COCO/Flickr30k. We report results using the COCO captioning evaluation toolkit [71], which reports the widely used automatic evaluation metrics, BLEU [231], METEOR [232], CIDEr [233] and SPICE [234].

We present our methods trained on different object detectors: Flickr and COCO. We compare our approach (referred to as NBT) to recently proposed Hard-Attention [34], ATT-FCN [198] and Adaptive [**lu**] on Flickr30k, and Att2in [212], Up-Down [37] on COCO. Since object detectors have not yet achieved near-perfect accuracies on these datasets, we also report the performance of our model under an oracle setting, where the ground truth object region and category is also provided during test time. (referred to as NBT$^{\text{oracle}}$) This can be viewed as the upper bound of our method when we have perfect object detectors.

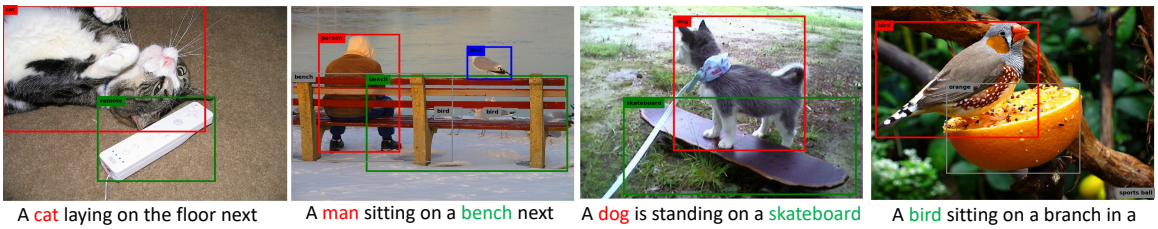


Figure 6.6: Generated captions and corresponding visual grounding regions for the robust image captioning task. "cat-remote", "man-bird", "dog-skateboard" and "orange-bird" are co-occurring categories excluded in the training split. First 3 columns show success and last column shows failure case (orange was not mentioned).

Method	BLEU4	METEOR	CIDEr	SPICE	Accuracy
Att2in [212]	31.5	24.6	90.6	17.7	39.0
Up-Down [37]	31.6	25.0	92.0	18.1	39.7
NBT	**31.7**	**25.2**	**94.1**	**18.3**	**42.4**
NBT$^{\text{oracle}}$	31.9	25.5	95.5	18.7	45.7

Table 6.3: Performance on the test portion of the robust image captioning split on COCO dataset.

Table 6.1 shows results on the Flickr30k dataset. We see that our method achieves state of the art on all automatic evaluation metrics, outperforming the previous state-of-art model Adaptive [200] by 2.0 and 4.4 on BLEU4 and CIDEr. When using ground truth proposals, NBT$^{\text{oracle}}$ significantly outperforms previous methods, improving 5.1 on SPICE, which implies that our method could further benefit from improved object detectors.

Table 6.2 shows results on the COCO dataset. Our method outperforms 4 out of 5 automatic evaluation metrics compared to the state of the art [212, 200, 37] without using better visual features or directly optimizing the CIDEr metric. Interestingly, the NBT$^{\text{oracle}}$ has little improvement over NBT. We suspect the reason is that explicit ground truth annotation is absent for visual words. Our model can be further improved with explicit co-reference supervision where the ground truth location annotation of the visual word is provided. Fig. 6.5 shows qualitative results on both datasets. We see that our model learns to correctly identify the visual word, and ground it in image regions even under weak supervision (COCO). Our model is also robust to erroneous detections and produces correct captions (3rd column).

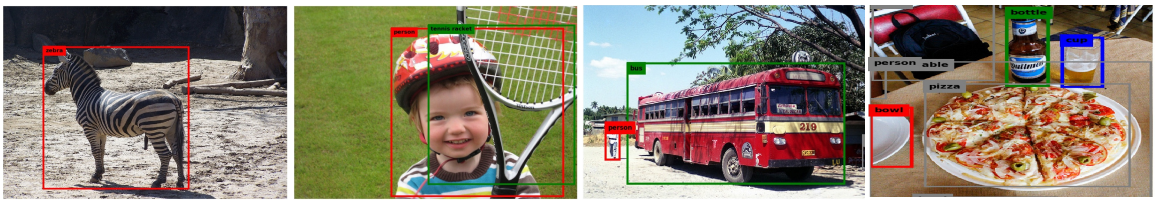


Figure 6.7: Generated captions and corresponding visual grounding regions for the novel object captioning task. "zebra", "tennis racket", "bus" and "pizza" are categories excluded in the training split. First 3 columns show success and last column shows a failure case.

	Out-of-Domain Test Data											
Method	bottle	bus	couch	microwave	pizza	racket	suitcase	zebra	Avg	SPICE	METEOR	CIDEr
DCC [222]	4.6	29.8	45.9	28.1	64.6	52.2	13.2	79.9	39.8	13.4	21.0	59.1
NOC [223]	17.8	68.8	25.6	24.7	69.3	68.1	39.9	89.0	49.1	-	21.4	-
C-LSTM [224]	29.7	74.4	38.8	27.8	68.2	70.3	44.8	91.4	55.7	-	23.0	-
Base+T4 [225]	16.3	67.8	48.2	29.7	77.2	57.1	49.9	85.7	54.0	15.9	23.3	77.9
NBT*+G	7.1	73.7	34.4	61.9	59.9	20.2	42.3	88.5	48.5	15.7	22.8	77.0
NBT†+G	14.0	74.8	42.8	63.7	74.4	19.0	44.5	92.0	53.2	16.6	23.9	84.0
NBT†+T1	36.2	77.7	43.9	65.8	70.3	19.8	51.2	93.7	57.3	16.7	23.9	85.7
NBT†+T2	**38.3**	**80.0**	**54.0**	**70.3**	**81.1**	**74.8**	**67.8**	**96.6**	**70.3**	**17.4**	**24.1**	**86.0**

Table 6.4: Evaluation of captions generated using the proposed method. G means greedy decoding, and T1−2 means using constrained beam search [225] with 1−2 top detected concepts. $*$ is the result using VGG-16 [103] and † is the result using ResNet-101.

6.4.2 Robust Image Captioning

To quantitatively evaluate image captioning models for novel scene compositions, we present a new split of the COCO dataset, called the robust-COCO split. This new split is created by re-organizing the train and val splits of the COCO dataset such that the distribution of co-occurring objects in train is different from test. We also present a new metric to evaluate grounding.

Robust split. To create the new split, we first identify entity words that belong to the 80 COCO object categories by following the same pre-processing procedure. For each image, we get a list of object categories that are mentioned in the caption. We then calculate the co-occurrence statistics for these 80 object categories. Starting from the least co-occurring category pairs, we greedily add them to the test set and ensure that for each category, at

least half the instances of each category are in the train set. As a result, there are sufficient examples from each category in train, but at test time we see novel compositions (pairs) of categories. Remaining images are assigned to the training set. The final split has 110,234/3,915/9,138 images in train/val/test respectively.

Evaluation metric. To evaluate visual grounding on the robust-COCO split, we want a metric that indicates whether or not a generated caption includes the new object combination. Common automatic evaluation metrics such as BLEU [231] and CIDEr [233] measure the overall sentence fluency. We also measure whether the generated caption contains the novel co-occurring categories that exist in the ground truth caption. A generated caption is deemed 100% accurate if it contains at least one mention of the *compositionally novel* category-pairs in any ground truth annotation that describe the image.

Results and analysis. We compare our method with state of the art Att2in [212] and Up-Down [37]. These are implemented using the open source implementation from [235] that can replicate results on Karpathy's split. We follow the experimental setting from [212] and train the model using the robust-COCO train set. Table 6.3 shows the results on the robust-COCO split. As we can see, all models perform worse on the robust-COCO split than the Karpathy's split by 2$\sim$3 points in general. Our method outperforms the previous state of the art methods on all metrics, outperforming Up-Down [37] by 2.7 on the proposed metric. The oracle setting (NBT$^{\text{oracle}}$) has consistent improvements on all metrics, improving 3.3 on the proposed metric.

Fig. 6.6 shows qualitative results on the robust image captioning task. Our model successfully produces a caption with novel compositions, such as "cat-remote", "man-bird" and "dog-skateboard" to describe the image. The last column shows failure cases where our model didn't select "orange" in the caption. We can force our model to produce a caption containing "orange" and "bird" using constrained beam search [225], further illustrated in Sec. 6.4.3.

6.4.3 Novel Object Captioning

Since our model directly fills the "slotted" caption template with the concept, it can seamlessly generate descriptions for out-of-domain images. We replicated an existing experimental design [222] on COCO which excludes all the image-sentence pairs that contain at least one of eight objects in COCO. The excluded objects are 'bottle', "bus", "couch", "microwave", "pizza", "racket", "suitcase" and "zebra". We follow the same splits for training, validation, and testing as in prior work [222]. We use Faster R-CNN in conjunction with ResNet-101 which is pre-trained on COCO train split as the detection model. Note that we do not pre-train the language model using COCO captions as in [222, 223, 224], and simply replace the novel object's word embedding with an existing one which belongs to the same super-category in COCO (e.g., bus $\leftarrow$ car).

Following [225], the test set is split into in-domain and out-of-domain subsets. We report F1 as in [222], which checks if the specific excluded object is mentioned in the generated caption. To evaluate the quality of the generated caption, we use SPICE, METEOR and CIDEr metrics and the scores on out-of-domain test data are macro-averaged across eight excluded categories. For consistency with previous work [37], the inverse document frequency statistics used by CIDEr are determined across the entire test set.

As illustrated in Table 6.4, simply using greedy decoding, our model (NBT*+G) can successfully caption novel concepts with minimum changes to the model. When using ResNet-101 and constrained beam search [225], our model significantly outperforms prior works under F1 scores, SPICE, METEOR, and CIDEr, across both out-of-domain and in-domain test data. Specifically, NBT†+T2 outperforms the previous state-of-art model C-LSTM by 14.6% on average F1 scores. From the category F1 scores, we can see that our model is less likely to select small objects, e.g. "bottle", "racket" when only using the greedy decoding. Since the visual words are grounded at the object-level, by using [225], our model was able to significantly boost the captioning performance on out-of-domain images. Fig. 6.7 shows qualitative novel object captioning results.

6.5 Discussion

In this chapter, we introduce Neural Baby Talk, a novel image captioning framework that produces natural language explicitly grounded in entities object detectors find in images. Our approach is a two-stage approach that first generates a hybrid template that contains a mix of words from a text vocabulary as well as slots corresponding to image regions. It then fills the slots based on categories recognized by object detectors in the image regions. We also introduce a robust image captioning split by re-organizing the train and val splits of the COCO dataset. Experimental results on standard, robust, and novel object image captioning tasks validate the effectiveness of our proposed approach.

CHAPTER 7

REASON ON SCENE GRAPH FOR VISUAL QUESTION GENERATION

In this chapter, I will discuss how we can use a scene graph representation of an image for visual question generation. Similar to the image captioning discussed above, visual question generation also involves compositing words and phrases which needs to be grounded on visual data. We assume a scenario that a visual system is put in an environment which contains some novel visual concepts. The goal is to enable the agent to ask questions to an oracle or a human so as to acquire the information about the novel concepts. To achieve this, the visual system needs maintain an internal memory of images on which objects are already known and which are not known. As a result, scene graph is a natural and good representation for this task. By representing the image as a symbolic scene graph, we build a visual question generation model which can reason on the scene graph to generate informative *and* unambiguous questions. The experimental results show that the proposed model outperforms other heuristic baselines on standard test set. We further evaluate the learned policy on some novel environments which are not seen during training, and find the policy generalize well to help the visual system to learn to recognize novel concepts.

7.1 Introduction

As the various artificial intelligence sub-fields (vision, language, reasoning) mature, we are beginning to see ambitious multi-disciplinary tasks being undertaken – at the intersection of vision-and-language (*e.g.*image captioning [210, 34, 15], visual question answering [236, 237], visual dialog [196]), vision-and-navigation [238, 239], and vision-language-and-navigation [240, 241, 242]. These tasks (and others) implicitly rely on the assumption that agent's visual recognition system is mature enough (*i.e.*can recognize scenes, objects, their attributes, relationships, *etc*.) to support these higher-level AI tasks.

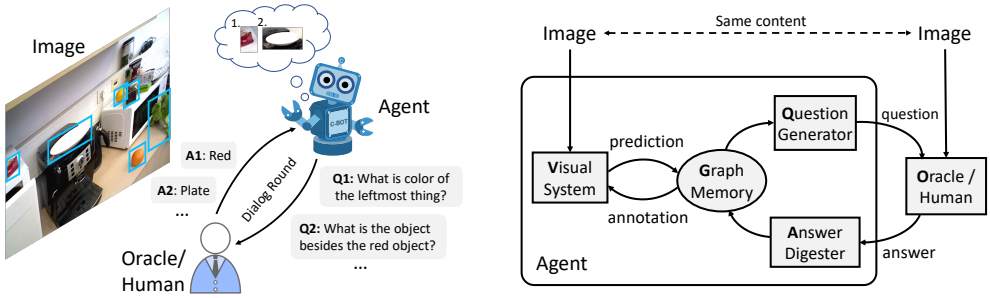


Figure 7.1: Left: an example scenario where the agent learns to recognize objects through a dialog with an Oracle. Right: the proposed framework contains a visual recognition module (to see), question generation policy (to ask), answer digester (to understand) and graph memory module (to memorize).

However, in an open world, it is inevitable that the agent will encounter some new visual content (new scenes, objects, attributes) that it has never seen before. In such cases, it is natural to consider whether the agent can simply 'ask' a human or an Oracle to identify the novel content and build visual classifiers on the fly. Note that this is a challenging task since the agent must (1) understand what it recognizes and what it does not, (2) formulate a valid, unambiguous and informative 'language' query (a question) to ask the Oracle, (3) derive the parameters of visual classifiers from the Oracle response and (4) leverage the updated visual classifiers to ask more clarified questions.

Towards this goal, we develop an agent with the ability to ask questions about an image to an Oracleand build visual classifiers based on the answers received. We call this ability – *visual curiosity*. Fig. 7.1 left illustrates this setup. Given an image, the agent's visual system generates object proposals (or candidate bounding boxes). The agent is confident about labels of some candidate boxes ('orange_fruit', 'lettuce'), but does not recognize the content in others. It generates a question What is the color of the leftmost object?. The Oracle responds with the answer red, which the agent uses to update its 'red' classifier. Furthermore, the agent uses the 'red object' as a referent in future rounds of dialog to acquire labels of other objects (What is the object besides the red object?).

One immediate question at this point may be – what is the relationship of this setup to active learning [243, 244, 245, 246]? A full discussion is available in Section 7.2, but in

short, our approach lies at the intersection of active learning and meta-learning – *i.e.*, instead of using a pre-specified active learning protocol, we *learn to actively learn* [247, 248, 249]. Specifically, we formulate this task as a reinforcement learning problem and *learn a policy* to ask questions to learn visual recognition. All components of our agent (illustrated in Fig. 7.1 right) – visual recognition module (to see), question generation policy (to ask), answer digester (to understand) and graph memory module (to memorize) – are learned entirely end-to-end to maximize the reward derived from the scene graph generated by the agent as a consequence of the dialog with the oracle.

Importantly, the question generation policy is disentangled from the visual recognition system and specifics of the environment (scenes). Consequently, we demonstrate a sort of 'double' generalization – our question generation policy generalizes to new environments *and* a new pair of eyes. Specifically, an agent trained on one set of environments (scenes) and with one particular visual recognition system is able to ask intelligent questions about new scenes when paired with a new visual recognition system (which may or may not recognize the same set of entities as the visual system during training).

Our results show that our agent – trained in a synthetic environment with a certain set of objects – learns new visual concepts significantly faster than several heuristic baselines when deployed in a synthetic environment with novel objects as well as in a more realistic environment.

In order to make progress on this challenging problem, we make a number of simplifying assumptions that are described in detail in Section 7.3 but highlighted here for completeness and full disclosure – we use templated questions with slots that are filled by the agent, and model only simple geometric relationships between object proposals (right, left, front, behind) that are trivial for the agent to extract from bounding box coordinates. Also, we assume the agent can localize objects in an image precisely. However, we believe the ideas and components of our work may generalize to more challenging scenarios in the future.

7.2 Background

Active Learning addresses the problem of selecting samples from an unlabeled set to be labeled by some oracle [243, 244, 245, 246]. Common selection criteria rely on heuristics, including entropy [250], expected model change [251], and boosting classifier margin [252]. Unlike traditional active learning, querying the oracle in our setting is not guaranteed to succeed; to gain a new label, agents must correctly refer to target objects when issuing queries to the Oracle. Further, our approach learns to collect labels efficeintly from end-to-end training rather than with predefined measures.

Meta Active Learning. Other recent work has also followed this *learning-to-active-learn* strategy [247, 248, 249], training meta-learning models to select sets of instances to be labeled in order to maximize performance of some target model trained on the selected set. As before, these models have direct access to the oracle labels. Further, these meta-learners are tightly coupled with their corresponding target model; being trained based on target model performance. In contrast, our approach is agnostic to the specific perception model.

Learning by Asking Questions. Mirsa *et al*.[253] present a learning-by-asking (LBA) framework for visual question answering (VQA). The main differences between our setting and LBA are two-fold: 1) We focus on learning a better visual system, not a better VQA model. Essentially, LBA is active learning (via language) for VQA, while our work is active learning (via language) to learn to see. 2) Our model decouples the visual system and question generation, which makes the learned question generator agnostic to different environments.

Teaching Robots via Language Interactions. Previous work in human-robot interaction focus on agents learning new concepts from speaking with human operators [46, 45, 44, 43, 254, 42]. Tellex *et al*.[42] present a generalized grounding graph framework based on the linguistic coreference. However, there question generation policy and vision system

are not designed to learn. Lütkebohle *et al*.[43] propose use language-interaction to solve ambiguity in the object references and for grasping commands. Thomason *et al*.[46] learn an active learning dialog policy for natural language grounding. Both [45] and [44] generate questions to continuous learn objects and visual properties. However, our goal is to learn a question generation policy that is distangled from the visual recognition system and specifics of the scenes, which enables both active learning and meta learning.

7.3 Learning to Ask Questions

As illustrated in Figure 7.1(b), there are four major components in our framework:

- **Visual System** $\mathcal{V}$ that localizes image regions with high 'objectness' (*i.e.*generates object proposals) and predicts their categories and attributes.
- **Question Generator** $\mathcal{Q}$ that identifies an object proposal to inquire about and generates a question based on graph memory to ask the Oracleabout its category or attribute;
- **Answer Digester** $\mathcal{D}$ that uses the Oracle's answer to update the graph memory for training visual system $\mathcal{V}$to recognize the contents for future images;
- **Graph Memory** that is a semantic graph representation connecting the other three components.

Graph Representation. In our work, the agent's graph memory is the underlying datastructure connecting all other components; thus, we describe it first. It captures information about the image that the agent has gathered from the Oracle. For an image I, $G = (V, E)$ denotes a directed graph where the nodes V correspond to the object proposals (with $|V| = K$), and edges E correspond to the relationships between proposals. Let denote a set of visual attributes (*e.g.*, object category, object shape) on object proposals. For an attribute concept $a \in$, n_a denotes the number of states for the concept a (*e.g.*concept

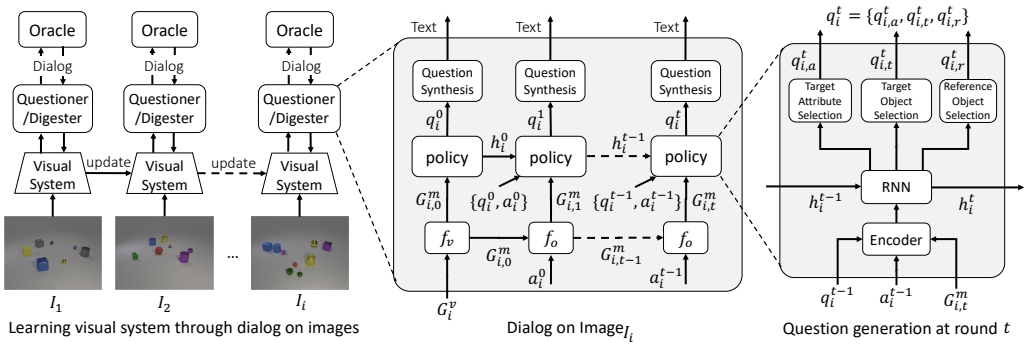


Figure 7.2: We simulate an agent observes a sequence of images, and interacts with the Oracle through dialogs to update its visual system (left). On each image, the agent asks a number of questions and gets responses from the Oracle (middle). For each question, the agent takes the history and the current graph memory as inputs and fills the question templates recurrently to compose a question (right).

color can be 'red', 'blue', 'green', etc). Let Δ^n be a n-simplex. Then, $\boldsymbol{p}^a \in \Delta^{n_a}$ denotes a probability distribution over attribute states for attribute concept a. Similarly, n_r denotes the number of spatial relationships and $\boldsymbol{p}^r \in \Delta^{n_r}$ denotes a probability distribution over these relationships. Besides these distributions, each object proposal has a spatial location $\boldsymbol{l}$. We can then write the nodes of the graph as $V = \{(\boldsymbol{l}_k, \boldsymbol{p}_k^{a_1}, \ldots, \boldsymbol{p}_k^{a_{||}})\}_{k=1}^K$ and the edges as $E = \{p^r_{i \to j} | i, j \in [K], i \neq j\}$. In this work, the spatial relationships (left, right, behind, front) between object proposals are trivially recognizable from bounding box coordinates such that $p^r_{i \to j}$ are always delta functions. As such, we drop them from the graph notation for simplicity – writing $G^m = \{(\boldsymbol{l}_k, \boldsymbol{p}_k^{a_1}, \ldots, \boldsymbol{p}_k^{a_{||}})\}_{k=1}^K$. Besides the agent's graph memory, the agent also predicts a scene graph from an image using the visual recognition module $\mathcal{V}$. We denote this graph as $G^v = \{(\boldsymbol{l}_k, \boldsymbol{v}_k^{a_1}, \ldots, \boldsymbol{v}_k^{a_{||}})\}_{k=1}^K$ where $\boldsymbol{v}^a \in \Delta^{n_a}$. Likewise, the oracle $\mathcal{O}$has an oracle graph $G^o = \{(\boldsymbol{l}_k, \boldsymbol{o}_k^{a_1}, \ldots, \boldsymbol{o}_k^{a_{||}})\}_{k=1}^{K^*}$ corresponding to the ground-truth scene graph with K^* objects. We use G_i^m, G_i^v and G_i^o to denote these graph representations for image I_i.

Environment Setup. To mimic the scenario of an agent traversing a novel environment while being instructed by a human about the world around it, we formalize our learning setup as a Markov Decision Process (MDP) over a series of image grounded

dialogs. Specifically, an episode consists of multi-round dialogs about a sequence of n images $I_1, ...I_n \in \mathcal{I}$. Our goal is to learn a good policy to discuss with the Oracle in turn one by one on these n images so as to learn a good visual system to recognize objects and attributes. If successful, each of these dialogs with the Oracle produces important annotations on which to train the visual system which in turn produces a stronger foundation for subsequent dialogs.

Rollout Process. This environmental setup can be represented by a recurrent process as depicted in Fig. 7.2 – the agent initializes the graph memory using predictions from its visual system, the agent holds a dialog with the oracle to update this memory, and then the information gained over the dialog is used to update the visual system before this process is repeated for the next image. More formally, assume we have access to a question generation policy π_q, and a visual system $\mathcal{V}$. Presented with the image I_i, the agent first extracts the visual graph G_i^v from the image with $\mathcal{V}$. Before beginning the dialog with the Oracle, the agent updates its initial graph memory $G_{i,0}^m$ based on G_i^v through a bottom-up update function $f_v(G_{i,0}^m, G_i^v)$. Then, the agent engages in a T round dialog with the oracle and maintains a sequence of graph memories $\{G_{i,0}^m, ..., G_{i,T}^m\}$ corresponding to its beliefs about the image I_i at each round. At round t, the agent proposes a question q_i^t using the policy π_q based on the whole dialog history $\mathcal{H}_i^t = \{G_{i,0}^m, q_i^1, a_i^1, G_{i,1}^m ..., q_i^{t-1}, a_i^{t-1}, G_{i,t-1}^m\}$. The Oracle receives the question q_i^t and generates an answer a_i^t based on oracle graph G_i^o. Upon receiving the answer, the agent updates its graph memory using the top-down update function $f_o(G_{i,t-1}^m, a_i^t)$.

At the end of dialog on I_i, the agent uses the final graph memory $G_{i,T}^m$ along with the accumulated graph memories $\{G_{1,T}^m, ..., G_{i-1,T}^m\}$ to update the visual system $\mathcal{V}$before going to the next image I_{i+1}. This recurrent procedure on n images is outlined in Alg. 3. At the end of this process, the agent produces a trained visual system that can recognize the objects and attributes in images. We will elaborate the detail of each component in following section.

Algorithm 3 $Rollout(\{I_1, ..., I_n\}, T)$: MDP rollout process on n images.

Inputs: Image sequence $\{I_1, ..., I_n\}$; Dialog budget T; Question generation policy π_q
Outputs: Visual system $\mathcal{V}$; Rewards $\{r_1^{1,T}, ..., r_n^{1,T}\}$

Initialize $G^o_{\{1,...,n\}}$ with ground truth, $G^m_{\{1,...,n\}}$ with uniform distribution
for $i \in [1 \cdots n]$ **do**
  $G_i^v \leftarrow \mathcal{V}(I_i)$ {Extract visual graph from I_i}
  $G^m_{i,0} \leftarrow f_v(G^m_{i,0}, G^v_i)$ {Initialize graph memory with visual graph}
  for $t \in [1 \cdots T]$ **do**
    $q_i^t \leftarrow \pi_q(\mathcal{H}_i^t)$ {Generate question}
    $a_i^t \leftarrow O(q_i^t, G_i^o)$ {Oracle answers the question}
    $G^m_{i,t} \leftarrow f_o(G^m_{i,t-1}, a_i^t)$ {Update graph memory with answers}
  end for
  Train $\mathcal{V}$ with $[G^m_{1,T} \cdots G^m_{i,T}]$ {Train visual system with graph memories}
end for=0

7.3.1 Model

We elaborate on each of the main components of our model in this section.

Question Generator $\mathcal{Q}$. In order to produce queries to Oracle that are informative, the agent selects from a set of template questions – filling in information from the graph memory. Inspired by [20], each template is associated with a functional program that operates on the oracle scene graph to get the Oracle answer. For example, the question 'What is the color of the metal object?' corresponds to query_color(unique(filter_material(metal, scene))). Using these templates, the question generation is equivalent to selecting the objects and attributes about which to inquire. Specifically, the policy needs to determine which object attribute to ask about (*i.e.*, target attribute), which object to ask about (*i.e.*, target object), and if applicable which object to refer to (*i.e.*, reference object). For instance, for the image in Fig. 7.1, the generated question may be "What is the <*white*> <*object*> besides the <*red object*>". The target object and attribute is <*object*> and <*white*> respectively. The reference object is <*red object*>.

Our question generation policy π_q is implemented using a recurrent neural network (RNN). The memory provided by a recurrent policy is important for the agent to know which questions have already been asked and whether they were meaningful or not ac-

cording to the responses from Oracle. As illustrated in Fig. 7.2 right side, it consists of two components, target selection policy and reference selection policy. The first one determines which object and attribute to ask about. The second one determines whether to use a reference and which one to use if needed. These two policies share the low-level representations. Hence, we first elaborate the representation we use.

Representation. At the t-th dialog round on I_i, the question generator takes the question q_i^{t-1} and answer a_i^{t-1} from last round, and graph memory $G^m = \{(\boldsymbol{l}_k, \boldsymbol{p}_k^{a_1}, \ldots, \boldsymbol{p}_k^{a_{||}})\}_{k=1}^K$ as the input. Based on these three inputs, we compute:

- **Entropy of Graph Memory**: For object k and its attribute a, we compute $e_k^a = Entropy(\boldsymbol{p}_k^a)$. For the whole scene graph memory, we obtain the entropy tensor $K \times |\mathcal{A}| \times 1$;

- **Location Embedding**. For each object, we normalize its bounding box location $\boldsymbol{l}_k$ with image size and then use a two-layer MLP $(4-4-2)$ to embed it to two dimensions. For all K objects, the dimension is $K \times 2$. Afterward, we duplicate it for all attribute concepts, and thus obtain a tensor $K \times |\mathcal{A}| \times 2$;

- **Target at last round**: For each of K objects, we use one-hot tensor to encode which target object and which attribute the agent pointed to at last dialog round. Hence, the dimension is $K \times |\mathcal{A}| \times 1$.

- **Reference at last round**: We use another one-hot vector to encode which reference object the agent pointed, and the dimension is $K \times 1$. We use another one-hot vector $K \times 1$ to record whether the agent use a reference object or not. If the agent does not use reference object, then the vector becomes zero vector. Similarly, We combine them and duplicate it for all attribute concepts to $K \times |\mathcal{A}| \times 2$;

- **Answer at last round**: We use one-hot vector to encode the answer from Oracle at last round. If the answer is valid, then we assign 1 to the target and reference

object slots; otherwise 0. As a result, we obtain $K \times |\mathcal{A}| \times 1$ and $K \times 1$ for target and reference, respectively. Afterward, we duplicate $K \times 1$ reference vector to $K \times |\mathcal{A}| \times 1$ and concatenate it with target tensor to obtain $K \times |\mathcal{A}| \times 2$.

Combining all the above signals, the final input to our question generator policy network at t-th dialog round is $x_t \in \mathcal{R}^{K \times |\mathcal{A}| \times 8}$. In our dataset, the number of attribute concept is 4. We replace $|\mathcal{A}|$ with 4 in the following for clarity. Given $x_t \in \mathcal{R}^{K \times 4 \times 8}$, we first reshaped it to $K \times 32$, where each row encode the graph memory and history for one object. Then we vectorize x_t to K vectors and feed them as a batch to a LSTM, obtaining new features $x_t^p \in \mathcal{R}^{K \times 64}$ by $h_t, c_t = lstm(h_{t-1}, c_{t-1}, x_t);\ x_t^p = h_t$, where h_{t-1} and c_{t-1} are the hidden state and cell memory from the lstm network at dialog round $(t-1)$. This x_t^p from the hidden state in lstm will be used in both target and reference policy.

Target policy. It is aimed at pointing the right target object and attribute concept to ask about. This can be completed by directly pointing one of $K \times 4$ slots. To propose meaningful target and reference objects, the context is important. In our work, we exploit graph convolutional layers [61] to pass the context information across different objects. Specifically, the GCN layer has the following typical formulation:

$$\boldsymbol{z}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} W \boldsymbol{z}_j^{(l)}\right) \tag{7.1}$$

where $\mathcal{N}(i)$ is the neighbors of node i; W is a learnable projection matrix; α_{ij} is the affinity between node i and j. In our model, we compute the affinity between two object nodes based on the spatial distance:

$$\alpha_{ij} = \alpha_{ji} = \exp\left(-\frac{d(i,j)}{d_{max}}\right) \tag{7.2}$$

where d_{max} is the maximal distance in all object pairs. Given this affinity matrix, we first reshape x_t^p to $(K \times 4) \times 16$ and pass the above tensor through two graph convolutional

layers to obtain $x_t^{tar} \in \mathcal{R}^{(K\times 4)\times 16}$. Then we pass x_t^{tar} through two-layer MLP (16-16-1) to obtain $(K \times 4)$ scores, and further a softmax layer to obtain a probability distribution $\boldsymbol{p}_t^{tar} = Softmax(mlp(x_t^{tar}))$. Besides the head for action, we have another head to compute the value. We simply perform average pooling for x_t^{tar}) and also pass it to two-layer MLP to obtain the value for each of the object nodes. At end, to select the target object and attribute, we use an epsilon greedy sampling ($\epsilon = 0.1$) strategy to choose one entry during training and choose the maximal one during testing.

Reference policy. It is aimed at determining whether to use reference object and which one if needed. It also takes x_t^p as input. To select the right reference, this policy needs to know which target object is selected. Suspect the k-th object is selected as the target, we take the corresponding k-th 1×64 vector, and replicate it, which is then concatenated with the remaining to obtain $x_t^{ref} \in \mathcal{R}^{(K-1)\times 128}$. To determine which object to select as the reference, we also use two graph convolutional layers to update x_t^{ref} to 64 dimension. Then, output is sent to a two-layer MLP (64-32-1) to obtain the $K-1$ dimensional scores over all candidate reference objects. Similar to the way used in target policy, the reference object is selected based epsilon greedy during training and the entry with maximal score during testing. Meanwhile, x_t^{ref} is fed into another two graph convolutional layers to 64-d, which are then average pooled to obtain a single 64-d vector. This vector is then sent to a two-layer MLP (64-32-1) to predict whether or not to use reference object. For both selecting reference and determining whether to use reference, we compute the value using x_t^{ref} as input.

Based on the above policies, we can deterministically compose a question with the corresponding template and ask it to the Oracle. We will introduce the details on the question template we used in our experiments below.

Oracle $\mathcal{O}$. Given the question from the agent, the Oracle answers the question by executing the functional program on the oracle graph G_i^o. However, the execution can fail in some cases. First, the question might be ambiguous. For example, the agent may ask

'What is the color of the sphere?' when there are multiple spheres in the image. Second, the question might be invalid. For example, it is invalid if the agent asks the same question as above when there are no spheres in the image. As a result, the Oracle has three types of responses to the agent: 1) the answer to the question, 2) 'ambiguous_question' and 3) 'invalid_question'. If the Oracle responds with an answer, *e.g.*, 'red' to the agent, the agent's graph memory will be updated, otherwise it will stay the same.

Updating the Graph Memory. The graph memory is updated from the bottom-up (via visual system $\mathcal{V}$) and top-down (via answer digester $\mathcal{D}$) with update functions f_v and f_o, respectively:

Bottom-Up $\boldsymbol{f_v}$: For object k and attribute a, its probability $\boldsymbol{p}_k^a$ is updated to a one-hot vector by setting its $\arg\max(\boldsymbol{v}_k^a)$-th entry to 1 and others to 0, if $\max(\boldsymbol{v}_k^a) > \tau_i$, where τ_i is a threshold that is annealed during the recurrent process, $\tau_i = \max(0.9, \exp(-i/n))$.

Top-Down $\boldsymbol{f_o}$: Suppose the agent asks about attribute a for object k, and the answer is the l-th category for that attribute concept, then the agent will update its graph memory by setting the corresponding l-th entry in $\boldsymbol{p}_k^a$ to 1, and others to 0.

Reward. A good question generator is one that asks meaningful questions to acquire knowledge about images from the Oracle. So we define the reward at each dialog round as:

$$r_i^t = R(G_{i,t-1}^m, G_{i,t}^m, G_i^o) = S(G_{i,t}^m, G_i^o) - S(G_{i,t-1}^m, G_i^o) \tag{7.3}$$

where $S(\cdot)$ measures the similarity between the graph memory and the oracle graph. The reward is the difference in similarities between the current time step and the previous one. The purpose is to learn an agent that asks meaningful questions at *each* time step so that it can recover as much information as possible within a budget of T questions.

7.3.2 Learning

The overall learning algorithm for the agent is summarized in Alg. 4. The policy π_q is updated at the end of each episode while the visual system is updated multiple times during

Algorithm 4 Learning to Ask Question to Learn Visual Recognition.

Inputs: Image sequence length n; Dialog budget T

0: Initialize parameters θ_π and θ_v for policy and visual system, respectively
0: **while** True **do**
0: Initialize parameters θ_v {Reset visual system at the beginning of episode}
0: $\boldsymbol{I} \leftarrow \{I_1, ..., I_n\} \sim \mathcal{E}$ {Sample n images from an environment}
0: $\mathcal{V}, \{r_1^{1,T}, ..., r_n^{1,T}\} \leftarrow Rollout(\boldsymbol{I}, T)$ {Rollout on n images with Algorithm 3}
0: Update θ_π using Eq. (7.5) {Train question generation policy}
0: **end while**=0

the inner $Rollout$. Recall that our goal is to learn a strong question generation policy that can ask useful questions across varied environments and differently skilled visual systems. To achieve this, we decouple the question policy from the visual system during training through two strategies: first, we introduce the semantic graph representation as an intermediate between perception and question generation; secondly, we reset the visual system to a random initialization at the beginning of each episode.

The visual system is trained inside the rollout process. At the end of dialogs on each image I_i, we append the graph memory to the history and use both for training the visual system. This training is a supervised learning task and the objective is:

$$\theta_v^* = \arg\min_{\theta_v} \sum_{I_i \in \boldsymbol{I}} \sum_{k=1}^{K_i} \sum_{a \in \mathcal{A}} -\boldsymbol{p}_k^a \cdot \log(\boldsymbol{v}_k^a) \tag{7.4}$$

where K_i is the number of object proposals in I_i; $[\cdot]$ denotes the inner product operator between two vectors. The above objective is targeted to minimize the cross entropy between the prediction of visual system $\boldsymbol{v}_k^a$ and the graph memory $\boldsymbol{p}_k^a$, which is a one-hot vector as mentioned before. We use standard gradient descent methods to optimize this objective.

We train the question generation policy π_q to recover as much information as possible from the Oracle in a limited budget, say T dialog rounds. To succeed, the agent must ask valid, unambiguous questions about uncertain object attributes. To train the policy π_q parametrized by θ_π, we consider maximizing the expected reward gained by the policy over

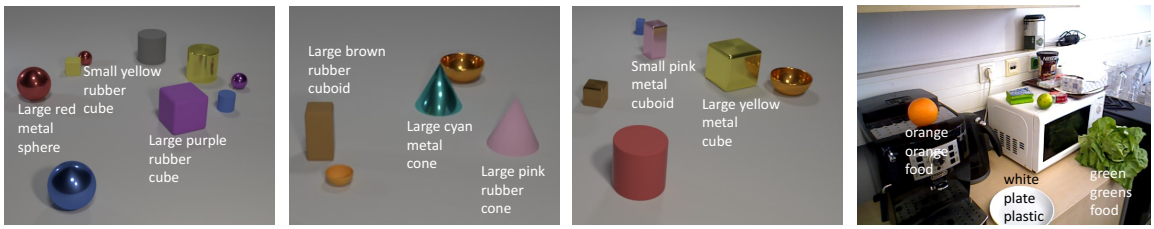


Figure 7.3: We use two types of datasets in our experiments. One is synthesized (left three columns) and one is a realistic dataset (right most). The synthesized one is further split to three sets, *standard*, *novel* and *mixed*.

episodes under environment $\mathcal{E}$,

$$\theta_\pi^* = \arg\max_{\theta_\pi} \; J(\theta_\pi) \;=\; \arg\max_{\theta_\pi} \; \mathbb{E}_{\mathcal{I}\sim\mathcal{E}}\mathbb{E}_{\pi_q}\left[\sum_{i=1}^{n}\sum_{t=1}^{T} r_i^t(q_i^t \sim \pi_q(\mathcal{H}_i^t))\right]. \tag{7.5}$$

In practice, we take a Monte Carlo estimate of this expected reward – sampling a sequence of images and questions throughout our dialogs – and use advantage actor-critic [255] (A2C) to train our agent.

7.4 Experiments

Recall that our goal is to learn visual curiosity, *i.e.*, a question generation policy that can intelligently ask questions to an Oracle and in doing so acquire meaningful information to train a visual recognition system. A successful agent should work well not only in the setting it was trained, but also in new environments that contain partially or entirely novel attributes and with different visual systems ranging in levels of competency. Moreover, as the visual system is decoupled from question generation, the agent should generalize well to new visual domains, e.g. from synthetic environments to realistic images. We evaluate our method for these qualities in the following experiments.

7.4.1 Dataset

We evaluate our question generation policy in both synthesized and realistic environments. Exemplar images are shown in Fig. 7.3. We generate the synthetic datasets using the same

API as [20]. Each image contains 5 to 10 objects each with four different attribute types (*shape*, *color*, *material*, and *size*). We construct three different datasets to test generalization; specifically, we generate:

Standard composed of objects from 3 shapes (cube, sphere, cylinder), 6 colors (gray, red, blue, green, yellow, purple), 2 materials (rubber, metal), and 2 sizes (large, small).

Novel consisting of objects from 3 novel shapes (cuboid, bowl, cone) and 4 new colors (pink, brown, cyan, orange) not present in *Standard*; however, materials and sizes are the same. The goal is to check generalization to novel attribute values.

Mixed which contains objects from all 6 shapes, 10 colors, 2 materials and 2 sizes from both *standard* and *novel* splits. This is used to test the generalization ability on complex scenes where some attributes are known and others are not.

We synthesize 1800 images which we split 900/300/600 for train, val, and test respectively. The *standard* train and val sets are used to train the agent policy, and the *standard*, *novel*, and *mixed* test sets are used for evaluation. For the realistic dataset, we use the images and bounding boxes from the Autonomous Robot Indoor Dataset (ARID) [256]. It contains 153 objects from 51 categories. We further annotated each object with one of 6 different materials and one of 11 colors. The agent trained on the synthetic *standard* split is also evaluated on this dataset.

7.4.2 Metrics and Baselines

For evaluation, we split each test set into 12 folds, each containing 50 images (i.e. a single episode sequence). We run the learned agent on each fold and evaluate two metrics:

Graph Recovery. We measure the correctness of the agent's graph memory. This measures how informative the agent's questions were. We compute the graph memory's recall with respect to the ground truth as the percentage of correctly predicted attributes. We report the average graph recall across testing folds at dialog round K as R@K. We also report the area under this curve as AUC.

Table 7.1: Graph recovery performance (i.e., quality of questions asked) on the *Standard*, *Novel*, and *Mixed* test sets for agents trained on *Standard*.

	Standard				*Novel*				*Mixed*			
Model	R@10	R@20	R@50	AUC	R@10	R@20	R@50	AUC	R@10	R@20	R@50	AUC
Random	28.3	36.5	59.4	0.41	23.4	31.2	54.0	0.36	27.0	37.3	63.2	0.43
Entropy	29.5	39.1	65.5	0.44	28.3	36.3	61.9	0.42	29.9	40.7	70.7	0.47
Entropy+Context	38.0	52.5	67.1	0.52	35.2	46.5	59.5	0.46	38.5	49.9	66.4	0.52
Our model	**42.1**	**59.1**	**89.3**	**0.63**	**43.3**	**58.4**	**88.9**	**0.64**	**42.9**	**60.1**	**90.3**	**0.64**
Our model w/o $\mathcal{V}$	25.8	50.6	84.1	0.55	25.5	50.0	85.2	0.55	26.8	51.7	87.2	0.57

Visual Recognition. To evaluate if a better question generator leads to a better visual system, we measure how well a visual system performs after being trained through the agent's interactions with the Oracle on the test fold. To do so, we report the average graph recall of the visual system predictions on the remaining folds.

We compare our proposed approach with three baselines:

- **Random.** This agent randomly samples question to ask i.e it selects the target attribute, target object, and reference objects uniformly at random.
- **Entropy.** An object/attribute with higher entropy (in the graph memory) is more likely to be chosen as the target. Likewise, objects with lower entropy are more likely to be references.
- **Entropy+Context.** The agent prefers to select uncertain (high entropy) object/attribute with reliable (low entropy) neighbors as reference objects. This way, the model prefers to ground questions on objects with low ambiguity.

For comparisons, we make no changes other than replacing our approach with the above baselines.

7.5 Results

Recall that we train on the *standard* train set and evaluate on the *standard*, *novel* and *mixed* test sets.

Questioner Graph Memory. We first compare graph memory recovery for different models. As seen in Table. 7.1, our approach consistently outperforms the baseline models by a significant margin across all three test settings. Further, the performance between *standard* and *novel/mixed* is similar, suggesting that our approach generalizes well to novel settings. The **Random** and **Entropy** baselines both struggle to propose unambiguous questions without the use of spatial context. The **Entropy+Context** model fairs better, but falls off later when the hand-crafted strategy fails to find unambiguous reference objects. Our model steadily improves over the entire dialog and has apparently found a much better question asking strategy that generalizes well across different environments.

Static Vision Ablation. We also evaluate an ablated version of our model (**Ours w/o**) which never updates its visual system $\mathcal{V}$. This model must ask questions essentially from 'scratch' without any bottom-up visual information. As shown in Table 7.1, the agent starts dialogs with significantly lower graph similarity scores than our full model; however, as the dialog proceeds, this agent performs similarly. This highlights that the agent has learned to ask informative questions and not to simply rely on steadily improving the visual system.

Visual System Performance. We report the visual recognition accuracies in Fig. 7.5(a-c). We take visual system checkpoints throughout the agent dialogs and evaluate them on the held out folds – tracking the evolution of the visual system through the agent's interactions with the Oracle. We find our approach outperforms the baselines significantly in all settings. This is somewhat unsurprising as question generation and visual system learning are naturally synergistic – with improvement of either leading to easier improvement in the other.

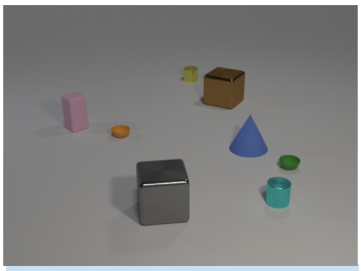
7.5.1 Transferring to Realistic Environment

Here, we apply the policy learned on the synthesized standard dataset to the realistic dataset. The episode length is also set to 50. In both situations, we observe significantly higher graph recalls for our model (86.2 R@50) than the baselines (56.7, 66.1, 55.5, for

random, entropy, and entropy+context respectively). More details are in Appendix. We also show the visual recognition curves in Fig. 7.5(d). As we can see, the learned question generator can flexibly adapt to the realistic dataset and the learned visual system outperforms other baselines by a large margin. These results imply that our model could perhaps be deployed on a real embodied agent and learn a visual system from traversing in an environment with a guide.

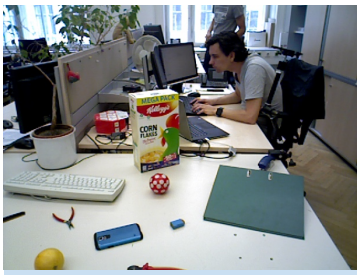
7.5.2 Qualitative Results

In Fig. 7.4, we show some qualitative results on both synthetic dataset and realistic dataset. Specifically, the questioner generation policy is trained on *Standard* train set, and then applied to test sets. Here, we display the first 16 rounds of dialog with oracle on three images, which are from *Mixed*, and ARID dataset. Our model learns to begin with zero-hop questions (blue), and followed with one-hop questions (green). Moreover, the learned question generation policy tends to repeatedly query one object until all the attributes are observed. When transferring to realistic environment, our method can successfully generalize to new objects and attributes, and ask meaningful questions. This verifies the effectiveness of our framework on disentangling the question generation policy and visual recognition system. We also find our model sometimes asks the ambiguous questions (red) which can not be answered by Oracle. The ambiguous questions can be either zero-hop question or one-hop question. When looking more closely, we find the ambiguous question is mainly caused by the unspecified target object, *e.g.*, "What size is the thing left of the small cyan shiny cylinder?", "There is a thing that ...". However, by taking the current graph memory and histories, the agent can successfully get rid of this soon after a few dialog rounds with the Oracle.



Q1: What is the closest thing made of? **A1**: metal
Q2: What shape is the closest object? **A2**: cylinder
Q3: What is the color of the closest object? **A3**: cyan
Q4: What size is the closest object? **A4**: small
Q5: What size is the thing left of the small cyan shiny cylinder? **A5**: __AMBIGUOUS__
Q6: What is the material of the rightmost thing? **A6**: rubber
Q7: What shape is the rightmost thing? **A7**: bowl
Q8: The rightmost thing has what color? **A8**: green
Q9: What is the size of the rightmost thing? **A9**: small
Q10 : There is a closest object to the left of the tiny green matte bowl; what is its material? **A10**: rubber
Q11: What shape is the closest matte object to the left of the tiny green matte bowl? **A11**: cone
Q12: There is a closest matte cone that is left of the tiny green matte bowl; what is its color? **A12**: blue
Q13: There is a closest blue rubber cone that is to the left of the small green rubber bowl; what is its size? **A13**: large
Q14: There is a thing that is on the left side of the small green rubber bowl; what is its material? **A14**: __AMBIGUOUS__
Q15: There is a closest object that is behind the large blue rubber cone; what is its material? **A15**: metal
Q16: There is a closest metal object that is behind the large blue matte cone; what is its shape? **A16**: cube

Q1: What is the shape of the farthest thing? **A1**: ball
Q2: What material is the farthest object? **A2**: plastic
Q3: What is the color of the farthest object? **A3**: yellow
Q4: How big is the closest thing behind the thing? **A4**: __AMBIGUOUS__
Q5: What material is the leftmost thing? **A5**: food
Q6: There is a leftmost object; what shape is it? **A6**: potato
Q7: The leftmost object is what color? **A7**: brown
Q8: What is the closest thing that is in front of the yellow plastic ball made of? **A8**: paper
Q9: What shape is the closest thing that is in front of the yellow plastic ball? **A9**: cereal
Q10: The closest paper cereal in front of the yellow plastic ball is what color? **A10**: red
Q11: There is a object that is in front of the yellow plastic ball; what is its material? **A11**: paper
Q12: What material is the closest object in front of the brown food potato? **A12**: food
Q13: The closest food thing that is in front of the food brown food potato has what shape? **A13**: onion
Q14: What color is the closest food onion that is in front of the brown food potato? **A14**: white
Q15: What is the object to the right of the food brown food potato made of? **A15**: __AMBIGUOUS__
Q16: There is a object that is behind the food white food onion; what is its material? **A16** __AMBIGUOUS__

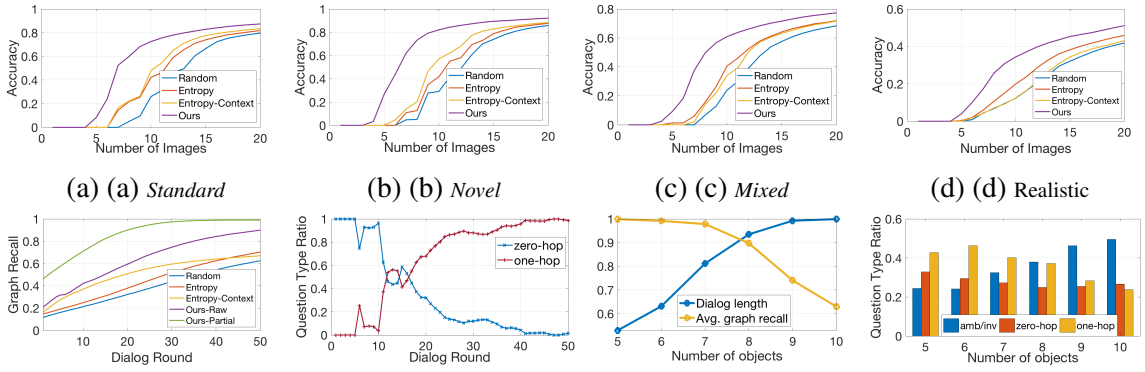


Q1: What is the rightmost thing made of? **A1**: plastic
Q2: There is a rightmost object; what shape is it? **A2**: stapler
Q3: The rightmost object is what color? **A3**: blue
Q4: There is a thing that is left of the blue plastic stapler; what size is it? **A4**: __AMBIGUOUS__
Q5: What is the object made of? **A5**: __AMBIGUOUS__
Q6: What material is the thing? **A6**: __AMBIGUOUS__
Q7: What material is the leftmost thing? **A7**: food
Q8: The leftmost thing is what shape? **A8**: orange
Q9: The leftmost thing is what color? **A9**: yellow
Q10: What material is the closest object right of the yellow food orange? **A10**: plastic
Q11: There is a closest plastic thing that is on the right side of the yellow food orange; what shape is it? **A11**: cell
Q12: There is a closest plastic cell right of the yellow food orange; what color is it? **A12**: black
Q13: There is a thing on the right side of the yellow food orange; what is it made of? **A13**: __AMBIGUOUS__
Q14: What is the material of the closest thing right of the black plastic cell? **A14**: plastic
Q15: The closest plastic thing that is right of the black plastic cell has what shape? **A15**: rubber
Q16: The closest plastic rubber that is right of the black plastic cell is what color? **A16**: blue

Figure 7.4: Dialogs with Oracle on *Mixed* synthesized dataset (left) and ARID dataset (middle and right) based on the policy learned on *normal* synthesized dataset. Questions in blue, green and red background corresponds to one-hop, two-hop and ambiguous questions respectively.

7.5.3 Inspecting the Question Generator

Questioner starting with partially learned visual system. We investigate how the question generator behaves on the *Mixture* set if its initial visual system can already recognize



(a) (a) *Standard* (b) (b) *Novel* (c) (c) *Mixed* (d) (d) Realistic

(e) (e) (f) (f) (g) (g) (h) (h)

Figure 7.5: Top Row: visual recognition accuracy curves against dialog round on different test sets. Bottom Row: Inspecting different aspects of question generation.

some of the attributes, *i.e.*, those in the *standard* set. In this case, the agent needs to ask about the remaining unknown attributes. In Fig. 7.5(e), we show average graph recalls on the *Mixture* test set. We find the average graph recall for this ablation (Ours-Partial) starts from a much higher graph recall, and continues to increase to almost 1 – indicating that the learned policy can also generalize well to partially trained visual systems. This is a promising result showing that the agent can leverage known visual concepts when learning about new ones and integrate additional visual systems seamlessly.

Question type against dialog round. We run our question generator on all *Standard* test images individually without updating the visual system. As shown in Fig. 7.5(f), it proposes more zero-hop questions at the beginning and then transitions to one-hop questions. This demonstrates that our model has learned an efficient strategy that asks questions about directly referable objects (e.g. leftmost) first and then objects that require referring to other (known) objects.

Questioner behavior with varying number of objects. We explore how the number of objects in images affect the question generation behavior. We separately evaluate the learned policy on images with varying number of objects. As shown in Fig. 7.5(g), the average graph recall on images decreases and the relative dialog length (divided by maximal length 50, so can be plotted from 0 to 1) increases when there are more objects in

the images. In Fig. 7.5(h), we can see there are fewer unambiguous/valid questions when the number of objects increases – implying that greater numbers of objects increases the difficulty for the questioner. However, our model can still perform well. As shown in Fig. 7.5(g), our approach still achieves an average graph recall of 89.75 with 8 objects present.

7.6 Discussion

In this chapter, we introduce a new setting *learning visual curiosity*, where an agent *learns* to ask questions to learn visual recognition. This is a challenging task where the agent needs to understand what it recognizes in an image and formulate language queries to the Oracle that are both unambiguous and informative. We use a graph memory to decouple the visual system and question generator. As a result, we demonstrate "double" generalization – we show that the learnt policy to ask informative questions generalizes to new environments as well as to a new visual system. We experimentally demonstrate that a policy learnt on a synthetic set of objects generalizes to novel objects, to mixture of novel objects and attributes, as well as to a realistic dataset – significantly better than strong baselines. This ability to learn about new objects and attributes by interacting with an Oracle is key to agents that operate in realistic open world settings.

7.7 Limitations and Future Directions

As we mentioned early, our work is an initial step towards learning visual curiosity. To start on this challenging task, we've made a number of simplifying assumptions that future work could soften. For instance, extending the model to more complicated visual scenarios where object proposal systems might be error-prone. In this case, the visual contents from the perspective of agent and Oracle are different from each other, which make the questions more ambiguous or confusing to the Oracle. One way to address this is empower the Oracle the visual curiosity ability as well, including answering clarifying questions to

the agent. Another extension is considering richer sets of relationships between objects, and enable the agent to learn about relationships as well during the interaction with Oracle. Further, models could be extended to operate on non-templated dialog exchange, *i.e.*, natural questions from agent and natural answers from Oracle. At last, in the current setting, we assume the number of the attribute concepts is given. However, incorporating with incremental learning to grow the attribute space over time would also be an interesting future direction to explore.

CHAPTER 8
CONCLUSION AND FUTURE WORK

In this thesis, we have discussed different ways to incorporate the structure prior to different vision tasks and its combination with language. By leveraging the dataset-level structure in an image set, we proposed an effective deep clustering method for image clustering and representation learning. At the image-level, we proposed to leverage the natural sparsity of scene structure for a better scene graph generation. At the object-level, we explore a way to enable the communication across different filters in a convolutional layer to learn structure-aware representations for image classification. Using the same inductive bias, we exploited the structure in a single image to propose a layer-wise image generation model. Finally, we utilize the structured representation of an image to bridge visual understanding and reasoning for image captioning and visual question generation. Our extensive experiments have shown that leveraging the structure prior in the visual data and textual data can not only improve the model performance but also achieve plausible groundingness and generalization ability.

Though we have demonstrated the advantages of leveraging structure prior in visual and textual data for various tasks, there are still some challenges ahead. First, the visual system is still far from satisfactory for structured visual understanding. As we observed from the experiments, the model performance for object detection, relationship detection drop drastically when the number of categories increase, not to mention the long-tail distribution in the data. Hence, how to develop a reliable visual understanding model which can scale to a large number of categories is still an open question. Second, recent vision models usually rely on a large amount of high-quality training data, which we cannot always obtain in practice. In this case, we need to resort to some weakly supervised or unsupervised learning paradigms to learn a good visual representations. Though we have witnessed a

number of works along this direction, they typically underperform supervised methods on a large portion of vision tasks. Hence, it is urgent to develop some unsupervised methods to learn structured visual representations that can generalize well across different vision tasks. Third, human intelligence is far more than visual perception. We interact with our surroundings through multiple modalities, like vision, language and action. We also have the skills to do causal reasoning, active learning, *etc*. How to extend a model to an agent that can interact with the environment and humans flexibly will be a promising direction to explore in the future.

Appendices

APPENDIX A

APPENDIX FOR LEVERAGE DATASET-LEVEL STRUCTURE FOR IMAGE CLUSTERING AND REPRESENTATION LEARNING

A.1 Affinity Measure for Clusters

In our work, we employ the affinity measure in [143]

$$\begin{aligned}\boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C}_j) &= \boldsymbol{\mathcal{A}}(\mathcal{C}_j \to \mathcal{C}_i) + \boldsymbol{\mathcal{A}}(\mathcal{C}_i \to \mathcal{C}_j) \\ &= \frac{1}{|\mathcal{C}_i|^2}\mathbf{1}_{|\mathcal{C}_i|}^{\boldsymbol{T}}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_j}\boldsymbol{W}_{\mathcal{C}_j,\mathcal{C}_i}\mathbf{1}_{|\mathcal{C}_i|} \\ &+ \frac{1}{|\mathcal{C}_j|^2}\mathbf{1}_{|\mathcal{C}_j|}^{\boldsymbol{T}}\boldsymbol{W}_{\mathcal{C}_j,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_j}\mathbf{1}_{|\mathcal{C}_j|}\end{aligned} \tag{A.1}$$

where $\boldsymbol{W}$ is the affinity matrix for samples, and $\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_j} \in \mathbb{R}^{|\mathcal{C}_i|\times|\mathcal{C}_j|}$ is the submatrix in $\boldsymbol{W}$ pointing from samples in $\mathcal{C}_i$ to samples in $\mathcal{C}_j$, and $\boldsymbol{W}_{\mathcal{C}_j,\mathcal{C}_i} \in \mathbb{R}^{|\mathcal{C}_j|\times|\mathcal{C}_i|}$ is the one pointing from $\mathcal{C}_j$ to $\mathcal{C}_i$. $\mathbf{1}_{|\mathcal{C}_i|}$ and $\mathbf{1}_{|\mathcal{C}_j|}$ are two vectors with all $|\mathcal{C}_i|$ and $|\mathcal{C}_j|$ elements be 1, respectively. Therefore, we have $\boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C}_j) = \boldsymbol{\mathcal{A}}(\mathcal{C}_j, \mathcal{C}_i)$.

According to (A.1), we can derive

$$\boldsymbol{\mathcal{A}}((\mathcal{C}_m \cup \mathcal{C}_n) \to \mathcal{C}_i) = \boldsymbol{\mathcal{A}}(\mathcal{C}_m \to \mathcal{C}_i) + \boldsymbol{\mathcal{A}}(\mathcal{C}_n \to \mathcal{C}_i) \tag{A.2}$$

which has also been shown in [143]. Meanwhile,

$$\begin{aligned}&\boldsymbol{\mathcal{A}}(\mathcal{C}_i \to (\mathcal{C}_m \cup \mathcal{C}_n)) \\ &= \beta\mathbf{1}_{|\mathcal{C}_m|+|\mathcal{C}_n|}^{\boldsymbol{T}}\boldsymbol{W}_{\mathcal{C}_m\cup\mathcal{C}_n,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m\cup\mathcal{C}_n}\mathbf{1}_{|\mathcal{C}_m|+|\mathcal{C}_n|} \\ &= \beta\mathbf{1}_{|\mathcal{C}_m|}^{\boldsymbol{T}}\boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m}\mathbf{1}_{|\mathcal{C}_m|} + \beta\mathbf{1}_{|\mathcal{C}_n|}^{\boldsymbol{T}}\boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n}\mathbf{1}_{|\mathcal{C}_n|} \\ &+ \beta\mathbf{1}_{|\mathcal{C}_m|}^{\boldsymbol{T}}\boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n}\mathbf{1}_{|\mathcal{C}_n|} + \beta\mathbf{1}_{|\mathcal{C}_n|}^{\boldsymbol{T}}\boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m}\mathbf{1}_{|\mathcal{C}_m|}\end{aligned} \tag{A.3}$$

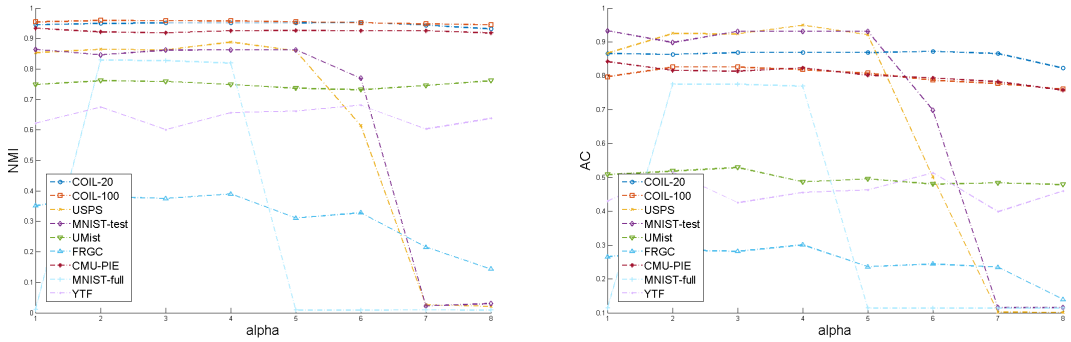


Figure A.1: Performance of agglomerative clustering with approximations. Left one is NMI metric, and right one is AC metric. The first column is without acceleration. For the other columns, $\alpha = \{-0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.5\}$.

where $\beta = 1/(|\mathcal{C}_m| + |\mathcal{C}_n|)^2$.

A.2 Approximated Affinity Measure

During agglomerative clustering, we need to re-compute the affinity between the merged cluster to all other clusters based on A.2 and A.3 repeatedly. It is simple to compute A.2. However, to get $\boldsymbol{\mathcal{A}}(\mathcal{C}_i \rightarrow (\mathcal{C}_m \cup \mathcal{C}_n))$, we need a lot of computations. These time costs become dominant and remarkable when we have a large-scale dataset. To accelerate the computations, we introduce an approximation method. At the right side of (A.3), we assume samples in $\mathcal{C}_m$ and $\mathcal{C}_n$ have similar affinities to $\mathcal{C}_i$. This assumption is mild because the condition to merge $\mathcal{C}_m$ and $\mathcal{C}_n$ is that they are similar to each other. In this case, the ratio between $\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m}\mathbf{1}_{|\mathcal{C}_m|}$ and $\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n}\mathbf{1}_{|\mathcal{C}_n|}$ is analogy to the ratio between the number of samples in two set, i.e.,

$$\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m}\mathbf{1}_{|\mathcal{C}_m|} = \frac{|\mathcal{C}_m|}{|\mathcal{C}_n|}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n}\mathbf{1}_{|\mathcal{C}_n|} \tag{A.4}$$

Based on (A.4), we have

$$\mathbf{1}^{\boldsymbol{T}}_{|\mathcal{C}_m|}\boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n}\mathbf{1}_{|\mathcal{C}_n|} = \frac{|\mathcal{C}_n|}{|\mathcal{C}_m|}\mathbf{1}^{\boldsymbol{T}}_{|\mathcal{C}_m|}\boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m}\mathbf{1}_{|\mathcal{C}_m|} \tag{A.5a}$$

$$\mathbf{1}_{|\mathcal{C}_n|}^{\boldsymbol{T}} \boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m} \mathbf{1}_{|\mathcal{C}_m|} = \frac{|\mathcal{C}_m|}{|\mathcal{C}_n|} \mathbf{1}_{|\mathcal{C}_n|}^{\boldsymbol{T}} \boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n} \mathbf{1}_{|\mathcal{C}_n|} \tag{A.5b}$$

As a result, we can re-formulate (A.3) to

$$\begin{aligned} &\boldsymbol{\mathcal{A}}(\mathcal{C}_i \to (\mathcal{C}_m \cup \mathcal{C}_n)) \\ &= \frac{1}{(|\mathcal{C}_m|^2 + |\mathcal{C}_m||\mathcal{C}_n|)} \mathbf{1}_{|\mathcal{C}_m|}^{\boldsymbol{T}} \boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m} \mathbf{1}_{|\mathcal{C}_m|} \\ &+ \frac{1}{(|\mathcal{C}_m||\mathcal{C}_n| + |\mathcal{C}_n|^2)} \mathbf{1}_{|\mathcal{C}_n|}^{\boldsymbol{T}} \boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n} \mathbf{1}_{|\mathcal{C}_n|} \end{aligned} \tag{A.6}$$

Therefore, we have

$$\begin{aligned} \boldsymbol{\mathcal{A}}(\mathcal{C}_i \to (\mathcal{C}_m \cup \mathcal{C}_n)) = &\frac{|\mathcal{C}_m|}{|\mathcal{C}_m| + |\mathcal{C}_n|} \boldsymbol{\mathcal{A}}(\mathcal{C}_i \to \mathcal{C}_m) \\ &+ \frac{|\mathcal{C}_n|}{|\mathcal{C}_m| + |\mathcal{C}_n|} \boldsymbol{\mathcal{A}}(\mathcal{C}_i \to \mathcal{C}_n) \end{aligned} \tag{A.7}$$

Consequently, we have

$$\begin{aligned} \boldsymbol{\mathcal{A}}(\mathcal{C}_m \cup \mathcal{C}_n, \mathcal{C}_i) = &\boldsymbol{\mathcal{A}}(\mathcal{C}_m \to \mathcal{C}_i) + \boldsymbol{\mathcal{A}}(\mathcal{C}_n \to \mathcal{C}_i) \\ &+ \frac{|\mathcal{C}_m|}{|\mathcal{C}_m| + |\mathcal{C}_n|} \boldsymbol{\mathcal{A}}(\mathcal{C}_i \to \mathcal{C}_m) \\ &+ \frac{|\mathcal{C}_n|}{|\mathcal{C}_m| + |\mathcal{C}_n|} \boldsymbol{\mathcal{A}}(\mathcal{C}_i \to \mathcal{C}_n) \end{aligned} \tag{A.8}$$

Above approximation provides us a potential way to reduce the computational complexity of agglomerative clustering. Though we computed $\boldsymbol{\mathcal{A}}(\mathcal{C}_i \to (\mathcal{C} \cup \mathcal{C}_n))$ based on Eq. (A.3) in all our experiments, we found the approximation version achieves analogy performance while costs much less time than the original one. We further simplify the computation by assuming a constant ratio α between the terms at the right side of Eq. (A.3):

$$\mathbf{1}_{|\mathcal{C}_m|}^{\boldsymbol{T}} \boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n} \mathbf{1}_{|\mathcal{C}_n|} = \alpha \mathbf{1}_{|\mathcal{C}_m|}^{\boldsymbol{T}} \boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m} \mathbf{1}_{|\mathcal{C}_m|} \tag{A.9a}$$

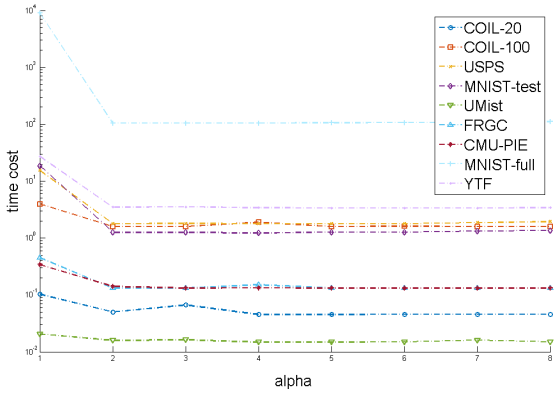


Figure A.2: Time cost for different values of α. The first column is the time cost without acceleration. For the other columns, $\alpha = \{-0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.5\}$.

$$\mathbf{1}_{|\mathcal{C}_n|}^{\boldsymbol{T}} \boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m} \mathbf{1}_{|\mathcal{C}_m|} = \alpha \mathbf{1}_{|\mathcal{C}_n|}^{\boldsymbol{T}} \boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n} \mathbf{1}_{|\mathcal{C}_n|} \tag{A.9b}$$

Based on above assumption,

$$\begin{aligned} \boldsymbol{\mathcal{A}}(\mathcal{C}_m \cup \mathcal{C}_n, \mathcal{C}_i) = & \boldsymbol{\mathcal{A}}(\mathcal{C}_m \to \mathcal{C}_i) + \boldsymbol{\mathcal{A}}(\mathcal{C}_n \to \mathcal{C}_i) \\ & + \frac{(1+\alpha)|\mathcal{C}_m|^2}{(|\mathcal{C}_m| + |\mathcal{C}_n|)^2} \boldsymbol{\mathcal{A}}(\mathcal{C}_i \to \mathcal{C}_m) \\ & + \frac{(1+\alpha)|\mathcal{C}_n|^2}{(|\mathcal{C}_m| + |\mathcal{C}_n|)^2} \boldsymbol{\mathcal{A}}(\mathcal{C}_i \to \mathcal{C}_n) \end{aligned} \tag{A.10}$$

We test various values for α, which are $\{-0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.5\}$. We show the quantitative comparison in Fig. A.1. We use image intensities as input to rule out all random factors. The original AC-GDL algorithm is used as the baseline. By conducting experiments on various datasets, we find a valid range $[-0.2, 0.1]$ for α which helps achieve analogous or even better performance to the one without acceleration. These results indicate that we may do not need to compute the explicit value of affinities to obtain equivalent level performance. Also, to measure how much time we can save by using our approximation, we compare the time cost between original AC-GDL algorithm and accelerated one in Fig. A.2. It is clear that our approximation algorithm has much lower computational complexity.

A.3 Cluster-based to Sample-based Loss

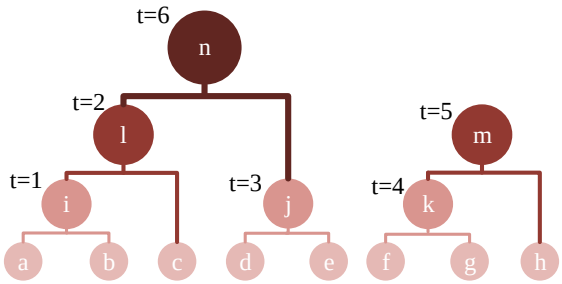


Figure A.3: A illustration of agglomerative clustering.

In this part, we explain how to convert cluster-based loss to sample-based loss. Because it depends on specific agglomerative clustering processes, we use a toy example in Fig. A.3 for illustration. We set K_c be 2 for simplicity. In Fig. A.3, there are six time steps, and thus $T = 6$. We assume they are in a single partial unrolled period. The leaf nodes represent single samples. For simplicity, we omit $\frac{\lambda}{K_c - 1}$ in (4.10), obtaining the overall loss

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{Y}}_*, \boldsymbol{I}) = -\sum_{t=1}^{6} \left(\lambda' \boldsymbol{\mathcal{A}}(\mathcal{C}_*^t, \mathcal{N}_{\mathcal{C}_*^t}^{K_c}[1]) - \boldsymbol{\mathcal{A}}(\mathcal{C}_*^t, \mathcal{N}_{\mathcal{C}_*^t}^{K_c}[2]) \right) \tag{A.11}$$

Given above loss function, we decompose it from first time step ($t = 1$) to the most recent time step ($t = 6$):

- **t=1**: $\mathcal{C}_*^1 = \mathcal{C}_a$, $\mathcal{N}_{\mathcal{C}_*^1}^2[1] = \mathcal{C}_b$ and $\mathcal{N}_{\mathcal{C}_*^1}^2[2] = \mathcal{C}_c$. We have

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}_*^1, I) = -\left(\lambda' \boldsymbol{\mathcal{A}}(\mathcal{C}_a, \mathcal{C}_b) - \boldsymbol{\mathcal{A}}(\mathcal{C}_a, \mathcal{C}_c)\right) \tag{A.12}$$

  Clearly, above is sample-based weighted triplet loss function, where samples $\mathcal{C}_a$ and $\mathcal{C}_b$ are positive pair and $\mathcal{C}_a$ and $\mathcal{C}_c$ are negative pair.

- **t=2**: $\mathcal{C}_*^2 = \mathcal{C}_i$, $\mathcal{N}_{\mathcal{C}_*^2}^2[1] = \mathcal{C}_c$ and $\mathcal{N}_{\mathcal{C}_*^2}^2[2] = \mathcal{C}_d$. We have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2\}, I) &= \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}_*^1, I) \\ &\quad - (\lambda' \boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C}_c) - \boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C}_d))\end{aligned} \tag{A.13}$$

Since $\mathcal{C}_i = \mathcal{C}_a \cup \mathcal{C}_b$, we base on Eq. (A.8) for approximation

$$\begin{aligned}\boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C}_c) &= \boldsymbol{\mathcal{A}}(\mathcal{C}_a \to \mathcal{C}_c) + \boldsymbol{\mathcal{A}}(\mathcal{C}_b \to \mathcal{C}_c) \\ &\quad + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_c \to \mathcal{C}_a) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_c \to \mathcal{C}_b)\end{aligned} \tag{A.14}$$

$$\begin{aligned}\boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C}_d) &= \boldsymbol{\mathcal{A}}(\mathcal{C}_a \to \mathcal{C}_d) + \boldsymbol{\mathcal{A}}(\mathcal{C}_b \to \mathcal{C}_d) \\ &\quad + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_a) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_b)\end{aligned} \tag{A.15}$$

Thus,

$$\begin{aligned}&\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2\}, I) \\ &= -\lambda' \boldsymbol{\mathcal{A}}(\mathcal{C}_a, \mathcal{C}_b) - (\lambda' - 1)\boldsymbol{\mathcal{A}}(\mathcal{C}_a \to \mathcal{C}_c) - \lambda' \boldsymbol{\mathcal{A}}(\mathcal{C}_b \to \mathcal{C}_c) \\ &- (\frac{\lambda'}{2} - 1)\boldsymbol{\mathcal{A}}(\mathcal{C}_c \to \mathcal{C}_a) - \frac{\lambda'}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_c \to \mathcal{C}_b) \\ &+ \boldsymbol{\mathcal{A}}(\mathcal{C}_a \to \mathcal{C}_d) + \boldsymbol{\mathcal{A}}(\mathcal{C}_b \to \mathcal{C}_d) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_a) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_b)\end{aligned} \tag{A.16}$$

At current time step, sample a, b and c belong to the same cluster $\mathcal{C}_l$, while sample d is from another cluster. (A.16) computes the sample-based weighted triplet loss for samples in $\mathcal{C}_l$ and sample d. Except for $\mathcal{C}_l$, the other clusters all have merely one sample. No need to compute triplet loss for them. It should be pointed out that λ' in above loss function should be not less than 2 so that the affinities for all pairs in $\mathcal{C}_l$ are enlarged.

- **t=3**: $\mathcal{C}_*^3 = \mathcal{C}_d$, $\mathcal{N}_{\mathcal{C}_*^3}^2[1] = \mathcal{C}_e$ and $\mathcal{N}_{\mathcal{C}_*^3}^2[2] = \mathcal{C}_f$. We have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2, \boldsymbol{y}_*^3\}, I) = \mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2\}, I) \\ - \lambda' \left(\boldsymbol{\mathcal{A}}(\mathcal{C}_d, \mathcal{C}_e) - \boldsymbol{\mathcal{A}}(\mathcal{C}_d, \mathcal{C}_f)\right)\end{aligned} \tag{A.17}$$

  Besides the loss $\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2\}, I)$ for $\mathcal{C}_l$, we also compute the loss for $\mathcal{C}_j$ in (A.17) because it contains two samples, *d* and *e*.

- **t=4**: $\mathcal{C}_*^4 = \mathcal{C}_f$, $\mathcal{N}_{\mathcal{C}_*^4}^2[1] = \mathcal{C}_g$ and $\mathcal{N}_{\mathcal{C}_*^4}^2[2] = \mathcal{C}_h$. We have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^4\}, I) = \mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2, \boldsymbol{y}_*^3\}, I) \\ - (\lambda' \boldsymbol{\mathcal{A}}(\mathcal{C}_f, \mathcal{C}_g) - \boldsymbol{\mathcal{A}}(\mathcal{C}_f, \mathcal{C}_h))\end{aligned} \tag{A.18}$$

  Here, we additionally compute the weighted triplet loss for cluster $\mathcal{C}_k$ since it contains two samples.

- **t=5**: $\mathcal{C}_*^5 = \mathcal{C}_k$, $\mathcal{N}_{\mathcal{C}_*^5}^2[1] = \mathcal{C}_h$ and $\mathcal{N}_{\mathcal{C}_*^5}^2[2] = \mathcal{C}_j$. We have

$$\begin{aligned}&\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^5\}, I) \\ &= \mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^4\}, I) - (\lambda' \boldsymbol{\mathcal{A}}(\mathcal{C}_k, \mathcal{C}_h) - \boldsymbol{\mathcal{A}}(\mathcal{C}_k, \mathcal{C}_j))\end{aligned} \tag{A.19}$$

  Because $\mathcal{C}_k = \mathcal{C}_f \cup \mathcal{C}_g$, we have

$$\begin{aligned}\boldsymbol{\mathcal{A}}(\mathcal{C}_k, \mathcal{C}_h) = \boldsymbol{\mathcal{A}}(\mathcal{C}_f \to \mathcal{C}_h) + \boldsymbol{\mathcal{A}}(\mathcal{C}_g \to \mathcal{C}_h) \\ + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_h \to \mathcal{C}_f) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_h \to \mathcal{C}_g)\end{aligned} \tag{A.20}$$

$$\begin{aligned}\boldsymbol{\mathcal{A}}(\mathcal{C}_k, \mathcal{C}_j) = \boldsymbol{\mathcal{A}}(\mathcal{C}_f \to \mathcal{C}_j) + \boldsymbol{\mathcal{A}}(\mathcal{C}_g \to \mathcal{C}_j) \\ + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_j \to \mathcal{C}_f) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_j \to \mathcal{C}_g)\end{aligned} \tag{A.21}$$

Since $\mathcal{C}_j = \mathcal{C}_d \cup \mathcal{C}_e$, we further transform above equation to

$$\begin{aligned}\boldsymbol{\mathcal{A}}(\mathcal{C}_k, \mathcal{C}_j) = &\frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_f \to \mathcal{C}_d) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_f \to \mathcal{C}_e) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_g \to \mathcal{C}_d) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_g \to \mathcal{C}_e) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_f) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_e \to \mathcal{C}_f) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_g) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_e \to \mathcal{C}_g)\end{aligned} \tag{A.22}$$

Similar to the relation between sample a and c at time steps $t = 1, 2$, sample f and h belong to the same cluster $\mathcal{C}_m$ at current time step while they are from different clusters at time step $t = 4$. Based on the approximation, the terms $\boldsymbol{\mathcal{A}}(\mathcal{C}_f \to \mathcal{C}_h)$ and $\boldsymbol{\mathcal{A}}(\mathcal{C}_h \to \mathcal{C}_f)$ in two time steps will be merged. As a result, the final loss is computed on intra-cluster pairs and inter-cluster pairs sampled from three clusters $\mathcal{C}_l$, $\mathcal{C}_j$ and $\mathcal{C}_m$.

- **t=6**: $\mathcal{C}_*^6 = \mathcal{C}_l$, $\mathcal{N}_{\mathcal{C}_*^6}^2[1] = \mathcal{C}_j$ and $\mathcal{N}_{\mathcal{C}_*^6}^2[2] = \mathcal{C}_m$. Thus

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^6\}, I) = &\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^5\}, I) \\ &- (\lambda' \boldsymbol{\mathcal{A}}(\mathcal{C}_l, \mathcal{C}_j) - \boldsymbol{\mathcal{A}}(\mathcal{C}_l, \mathcal{C}_m))\end{aligned} \tag{A.23}$$

Similar to the decomposition procedures above, both $\boldsymbol{\mathcal{A}}(\mathcal{C}_l, \mathcal{C}_j)$ and $\boldsymbol{\mathcal{A}}(\mathcal{C}_l, \mathcal{C}_m)$ can be transformed to sample-based affinities. Because $\mathcal{C}_l$ and $\mathcal{C}_j$ are regarded as different clusters previously, sample pairs from both of them are with positive weights in the loss function. However, it will be diminished by positive pairs (with negative weights) at current time step.

Though we use a toy example to show that the cluster-based loss can be transformed to sample-based loss above, the reduction is general to any possible agglomerative clustering processes because the loss for clusters at high-level can always be decomposed to the losses on clusters at low-level until it reaches to single samples. The difference among various

processes lies on the different weights associated with sample-based affinities. We should know that sample pairs from the same cluster may be with positive weights. One way to avoid this is increase λ'. In our implementation, we aim to increase affinities between samples from the same clusters, while decrease the affinities between samples from different clusters. And the clusters are determined by cluster ids at current step. Therefore, we assign a consistent weight γ to any affinities from the same cluster and 1 to any affinities from different clusters. Because we use SGD for batch optimization, the scales for affinities do not affect much on the performance. It is the signs affect much. Accordingly, at any given time step T, the overall loss is approximated to

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}_*^T, \boldsymbol{I}) = -\frac{\lambda}{K_c - 1}\sum_{i,j,k}(\gamma\boldsymbol{\mathcal{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \boldsymbol{\mathcal{A}}(\boldsymbol{x}_i, \boldsymbol{x}_k)) \tag{A.24}$$

Note that we replace $\boldsymbol{\mathcal{Y}}_*$ in (A.24) by $\boldsymbol{y}_*^T$ in (A.24) because it is merely determined by current $\boldsymbol{y}^T$, regardless of $\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^{T-1}\}$. As a result, we do not need to record $\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^{T-1}\}$. This simplifies the batch optimization for CNN. Concretely, given a sample $\boldsymbol{x}_i$, we randomly select a sample $\boldsymbol{x}_j$ which belongs to the same cluster, while select neighbours of $\boldsymbol{x}_i$ that from other clusters to be $\boldsymbol{x}_k$. To omit the case that $\boldsymbol{\mathcal{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is much larger than $\boldsymbol{\mathcal{A}}(\boldsymbol{x}_i, \boldsymbol{x}_k)$, we also add a margin threshold like the triplet loss function used in [144, 145].

A.4 Detailed CNN Architectures in our Paper

In this paper, the CNN architectures vary from dataset to dataset. As we mentioned in the main paper, we stacked different number of layers for different datasets so that the size of most top layer response map is about 10×10. In Table A.1, we list the architectures for the datasets used in our paper. "conv" means convolutional layer. "bn" means batch normalization layer. "wt-loss" means weighted triplet loss layer. ✓ means the layer is used, while $-$ means the layer is not used.

Table A.1: CNN architectures for different datasets in our paper.

Dataset	*COIL20*	*COIL100*	*USPS*	*MNIST-test*	*MNIST-full*	UMist	*FRGC*	*CMU-PIE*	*YTF*
conv1	✓	✓	✓	✓	✓	✓	✓	✓	✓
bn1	✓	✓	✓	✓	✓	✓	✓	✓	✓
relu1	✓	✓	✓	✓	✓	✓	✓	✓	✓
pool1	✓	✓	✓	✓	✓	✓	✓	✓	✓
conv2	✓	✓	–	✓	✓	✓	✓	✓	✓
bn2	✓	✓	–	✓	✓	✓	✓	✓	✓
relu2	✓	✓	–	✓	✓	✓	✓	✓	✓
pool2	✓	✓	–	–	–	✓	✓	✓	✓
conv3	✓	✓	–	–	–	✓	–	–	–
bn3	✓	✓	–	–	–	✓	–	–	–
relu3	✓	✓	–	–	–	✓	–	–	–
pool3	✓	✓	–	–	–	✓	–	–	–
conv4	✓	✓	–	–	–	–	–	–	–
bn4	✓	✓	–	–	–	–	–	–	–
relu4	✓	✓	–	–	–	–	–	–	–
pool4	✓	✓	–	–	–	–	–	–	–
ip1	✓	✓	✓	✓	✓	✓	✓	✓	✓
l2-norm	✓	✓	✓	✓	✓	✓	✓	✓	✓
wt-loss	✓	✓	✓	✓	✓	✓	✓	✓	✓

APPENDIX B

APPENDIX FOR REASON ON SCENE GRAPH FOR VISUAL QUESTION GENERATION

B.1 Question Templates

In our question templates, we introduce four attribute concepts size (<Z>), color (<C>, material (<M>), shape (<S>). We use <R> to depict the relation between two objects, which could be 'left', 'right', 'front' and 'behind'. Besides, we introduce the absolute spatial relationship <P> to depict the spatial location of one object proposal to the whole image. According to its location, it can be 'left-most', 'right-most', 'closest' 'farthest' or 'None' otherwise. Further, we use <L> to indicate whether the target object proposal is at the extreme location among all proposals that have the relationship <R> to its reference. It can be 'closest' if it is extreme, and 'None' otherwise. For clarification, we show two exemplar question templates below:

- "What shape is the <P> <Z> <C> <M> <S>?"
- "What size is <L> <Z> <C> <M> <S> that is <R> <Z> <C> <M> <S>?"

The question generator first points to the target object and reference object (if needed) and fill them into the above templates correspondingly. Based on the locations of target and reference objects, the relationship <R>, absolute location <P> and relative location <L> are manually inferred. Thus far, we can compose a unique questions which is then forward to Oracle side. Fig. B.2 and Fig. B.3 show the zero-hop and one-hop text and program templates on 4 attributes respectively.

B.2 Implementation Details

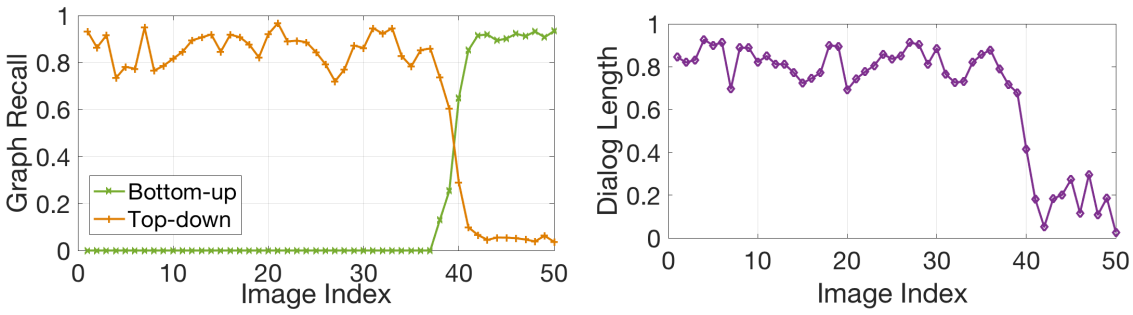
Visual system. We use Faster-RCNN [51] in conjunction with a pre-trained VGG16 [103] as the backbone of the visual system. We use the implemetnation open sourced in [73]. During training, the backbone is fixed, and we only learn the parameters for the four attribute classifiers (shape, color, size, material), which are two layer MLPs. We use the ground-truth bounding boxes on the agent side, since proposing object regions from the images is not our focus. In the future, we will try to use a region proposal network (RPN) to get the object proposals on the agent side. We are effectively assuming the the agent understands what constitutes an object, just not their names or attributes.

Question templates. We use zero-hop and one-hop question templates in our model. This is for two reasons: 1) they are enough to compose informative questions; 2) lower hop questions are more plausible to humans. Two simplified exemplar templates are: 1) Zero-hop: "What shape is the <*Some Object*>?"; 2) "What size is <*One Object*> that is <*Spatial Relation*> <*Another Object*>?"

During training, 100 images are sampled in an episode and the question generator is trained over 200 episodes. The visual system is updated with 50 gradient descent steps during each update. We start the visual system update once it accumulates $5 \times n_a$ annotations for attribute concept a. We use the Adam optimizer [228] for training the model. The learning rate starts from 1e-4 and decreases by a factor 0.99 after each image.

Table B.1: Graph recall on realistic test set with policy trained on synthetic *Standard* train set.

	Realistic			
Model	R@10	R@20	R@50	AUC
Random	20.1	27.5	56.7	0.33
Entropy	20.3	32.0	66.1	0.39
Entropy+Context	30.2	39.9	55.5	0.41
Ours model	**35.6**	**53.4**	**86.2**	**0.59**



(a) (b)

Figure B.1: Left: graph recall contributed bottom-up and top-down; Right: relative dialog length over time.

B.3 Graph recall from Bottom-up and Top-down

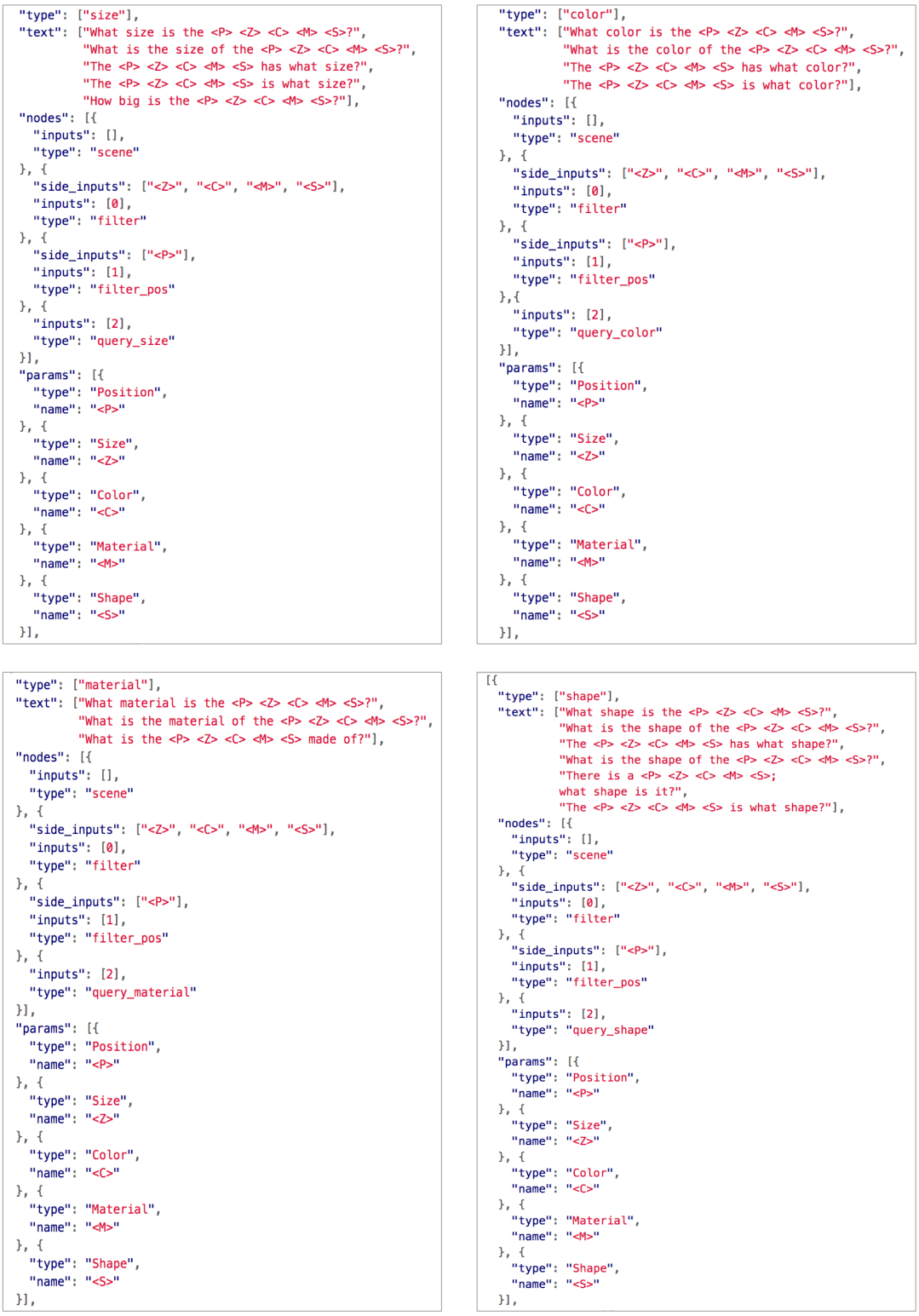
Recall that the graph memory is updated both by the visual system as well as information from the oracle. We investigate the contributions of these two factors to the graph recovery over the dialogs. As shown in Fig. B.1(a), as the dialog proceeds, the agent relies more on its visual system and less on interactions with Oracle. Since the graph memory is either updated bottom-up or top-down, we can easily measure their contributions by counting the number of entries updated by visual system and oracle in the graph memory. Also, as shown in Fig. B.1(b), the number of dialog rounds drops. This is a plausible behavior since we would not expect an intelligent agent to keep asking questions repeatedly after multiple interactions with Oracle.

B.4 Graph Recall on Realistic Environment

In Table B.1, we present the graph recalls for different methods on our collected realistic dataset. Clearly, our model outperforms all three baselines significantly. Though not being trained on the realistic environment, our questioner successfully propose meaningful questions to ask and get much higher graph recalls. Moreover, the numbers are comparable to those reported on synthetic test sets. **These numbers indicate that our model have a strong generalization ability across different environments.**

B.5 Attribute Annotations for ARID

For ARID dataset, we annotate three attribute concepts: object, color and material. Though the huge number of object categories than our synthetic dataset, our model trained on synthetic dataset generalizes well to this new environment. Fig. B.4 and Fig. B.5 show the example of an image and the associated scene graph on *mixture* synthesized dataset and ARID dataset, respectively.



```
"type": ["size"],
"text": ["What size is the <P> <Z> <C> <M> <S>?",
         "What is the size of the <P> <Z> <C> <M> <S>?",
         "The <P> <Z> <C> <M> <S> has what size?",
         "The <P> <Z> <C> <M> <S> is what size?",
         "How big is the <P> <Z> <C> <M> <S>?"],
"nodes": [{
  "inputs": [],
  "type": "scene"
}, {
  "side_inputs": ["<Z>", "<C>", "<M>", "<S>"],
  "inputs": [0],
  "type": "filter"
}, {
  "side_inputs": ["<P>"],
  "inputs": [1],
  "type": "filter_pos"
}, {
  "inputs": [2],
  "type": "query_size"
}],
"params": [{
  "type": "Position",
  "name": "<P>"
}, {
  "type": "Size",
  "name": "<Z>"
}, {
  "type": "Color",
  "name": "<C>"
}, {
  "type": "Material",
  "name": "<M>"
}, {
  "type": "Shape",
  "name": "<S>"
}],
```

```
"type": ["color"],
"text": ["What color is the <P> <Z> <C> <M> <S>?",
         "What is the color of the <P> <Z> <C> <M> <S>?",
         "The <P> <Z> <C> <M> <S> has what color?",
         "The <P> <Z> <C> <M> <S> is what color?"],
"nodes": [{
  "inputs": [],
  "type": "scene"
}, {
  "side_inputs": ["<Z>", "<C>", "<M>", "<S>"],
  "inputs": [0],
  "type": "filter"
}, {
  "side_inputs": ["<P>"],
  "inputs": [1],
  "type": "filter_pos"
},{
  "inputs": [2],
  "type": "query_color"
}],
"params": [{
  "type": "Position",
  "name": "<P>"
}, {
  "type": "Size",
  "name": "<Z>"
}, {
  "type": "Color",
  "name": "<C>"
}, {
  "type": "Material",
  "name": "<M>"
}, {
  "type": "Shape",
  "name": "<S>"
}],
```

```
"type": ["material"],
"text": ["What material is the <P> <Z> <C> <M> <S>?",
         "What is the material of the <P> <Z> <C> <M> <S>?",
         "What is the <P> <Z> <C> <M> <S> made of?"],
"nodes": [{
  "inputs": [],
  "type": "scene"
}, {
  "side_inputs": ["<Z>", "<C>", "<M>", "<S>"],
  "inputs": [0],
  "type": "filter"
}, {
  "side_inputs": ["<P>"],
  "inputs": [1],
  "type": "filter_pos"
}, {
  "inputs": [2],
  "type": "query_material"
}],
"params": [{
  "type": "Position",
  "name": "<P>"
}, {
  "type": "Size",
  "name": "<Z>"
}, {
  "type": "Color",
  "name": "<C>"
}, {
  "type": "Material",
  "name": "<M>"
}, {
  "type": "Shape",
  "name": "<S>"
}],
```

```
[{
  "type": ["shape"],
  "text": ["What shape is the <P> <Z> <C> <M> <S>?",
           "What is the shape of the <P> <Z> <C> <M> <S>?",
           "The <P> <Z> <C> <M> <S> has what shape?",
           "What is the shape of the <P> <Z> <C> <M> <S>?",
           "There is a <P> <Z> <C> <M> <S>;
           what shape is it?",
           "The <P> <Z> <C> <M> <S> is what shape?"],
  "nodes": [{
    "inputs": [],
    "type": "scene"
  }, {
    "side_inputs": ["<Z>", "<C>", "<M>", "<S>"],
    "inputs": [0],
    "type": "filter"
  }, {
    "side_inputs": ["<P>"],
    "inputs": [1],
    "type": "filter_pos"
  }, {
    "inputs": [2],
    "type": "query_shape"
  }],
  "params": [{
    "type": "Position",
    "name": "<P>"
  }, {
    "type": "Size",
    "name": "<Z>"
  }, {
    "type": "Color",
    "name": "<C>"
  }, {
    "type": "Material",
    "name": "<M>"
  }, {
    "type": "Shape",
    "name": "<S>"
  }],
```

Figure B.2: Zero-hop text and program templates on 4 attributes concepts (size, color, material, shape)

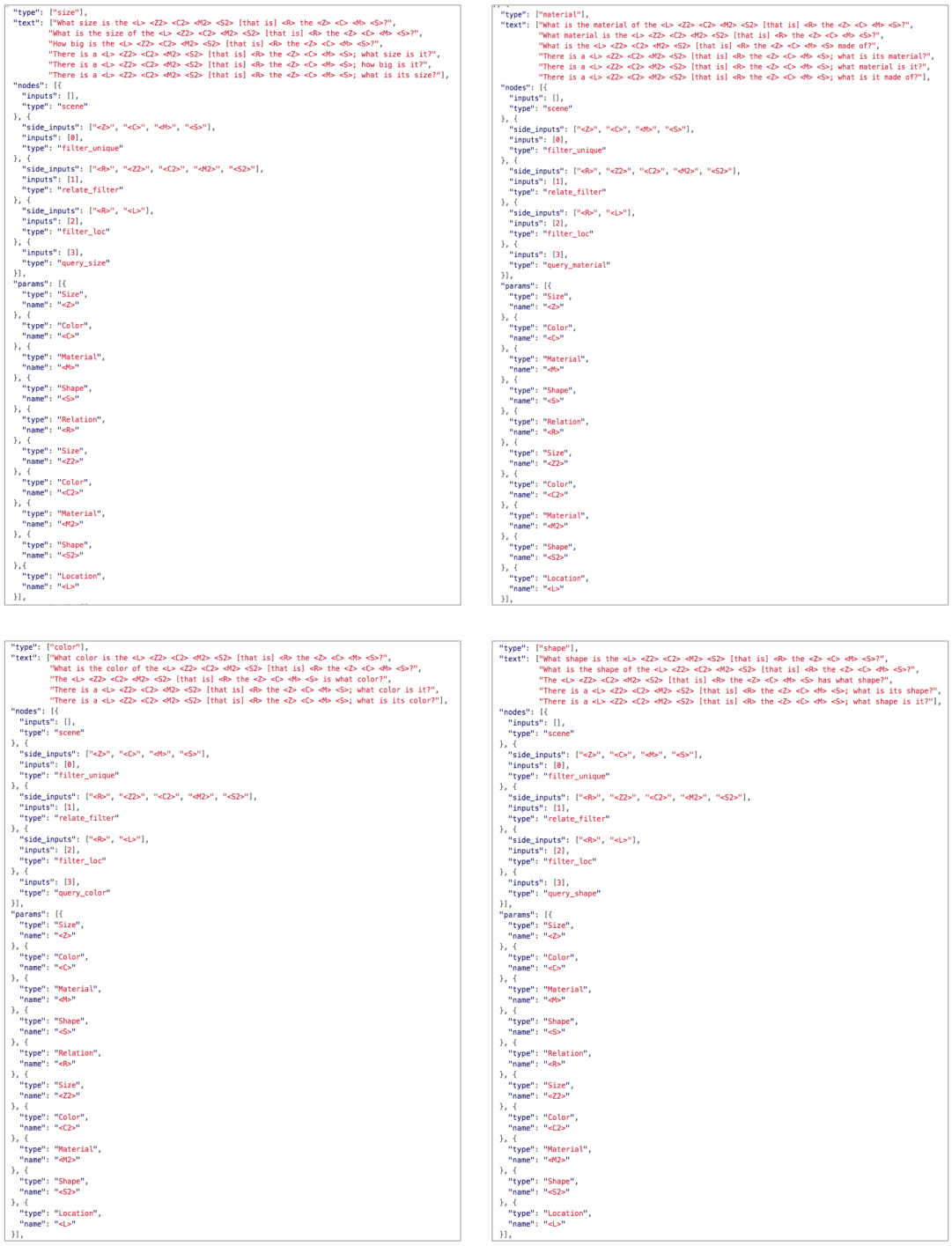
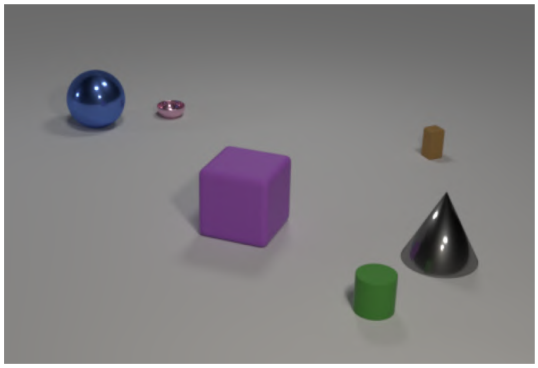


Figure B.3: One-hop text and program templates on 4 attributes concepts (size, color, material, shape)




```
"objects": [
  {
    "material": "metal",
    "bbox": [
      "347",
      "371",
      "371",
      "387"
    ],
    "size": "small",
    "shape": "bowl",
    "color": "pink"
  },
  {
    "material": "rubber",
    "bbox": [
      "520",
      "193",
      "558",
      "243"
    ],
    "size": "small",
    "shape": "cylinder",
    "color": "green"
  },
  {
    "material": "rubber",
    "bbox": [
      "580",
      "335",
      "600",
      "365"
    ],
    "size": "small",
    "shape": "SmoothRec",
    "color": "brown"
  },
  {
    "material": "rubber",
    "bbox": [
      "384",
      "258",
      "464",
      "346"
    ],
    "size": "large",
    "shape": "cube",
    "color": "purple"
  },
  {
    "material": "metal",
    "bbox": [
      "562",
      "230",
      "629",
      "306"
    ],
    "size": "large",
    "shape": "cone",
    "color": "gray"
  },
  {
    "material": "metal",
    "bbox": [
      "267",
      "363",
      "318",
      "414"
    ],
    "size": "large",
    "shape": "sphere",
    "color": "blue"
  }
```


Figure B.4: Example of an image and the associated scene graph on *mixture* synthesized dataset.


```
"objects": [
  {
    "color": "green",
    "material": "food",
    "bbox": [
      396.2573099415205,
      231.6608187134503,
      553.4645162692232,
      386.9165503124339
    ],
    "object": "greens"
  },
  {
    "color": "red",
    "material": "metal",
    "bbox": [
      337,
      184.68658497994187,
      367.80110617659136,
      229.42488757860679
    ],
    "object": "food"
  },
  {
    "color": "green",
    "material": "cloth",
    "bbox": [
      245.5460362316134,
      178.57835375501455,
      290.4152537472099,
      244.48074420773958
    ],
    "object": "sponge"
  },
  {
    "color": "green",
    "material": "food",
    "bbox": [
      313.4054874832849,
      190.43780500668603,
      352.58120642395875,
      228.05129317853203
    ],
    "object": "lime"
  },
  {
    "color": "white",
    "material": "glass",
    "bbox": [
      216.46783625730995,
      390,
      306.41945202998147,
      543.2625728564121
    ],
    "object": "plate"
  },
  {
    "color": "orange",
    "material": "food",
    "bbox": [
      14.239766081871345,
      208.06432748538012,
      91.90058479532163,
      289.9356725146199
    ],
    "object": "orange"
  }
```


Figure B.5: Example of an image and the associated scene graph on ARID dataset.

REFERENCES

[1] K. Gong, X. Liang, Y. Li, Y. Chen, M. Yang, and L. Lin, "Instance-level human parsing via part grouping network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 770–785.

[2] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1209–1218.

[3] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *arXiv preprint arXiv:1602.07332*, 2016.

[4] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2641–2649.

[5] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE conference on computer vision and pattern recognition*, IEEE, 2008, pp. 1–8.

[6] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 2010, pp. 2241–2248.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005.

[8] D. G. Lowe *et al.*, "Object recognition from local scale-invariant features.," in *iccv*, vol. 99, 1999, pp. 1150–1157.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] R. Girshick, "Fast r-cnn," in *CVPR*, 2015.

[11] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[12] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 618–626.

[13] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *CVPR*, 2015.

[14] Q. Wu, C. Shen, P. Wang, A. Dick, and A. van den Hengel, "Image captioning and visual question answering based on attributes and external knowledge," *PAMI*, 2017.

[15] J. Lu, J. Yang, D. Batra, and D. Parikh, "Neural baby talk," in *CVPR*, 2018.

[16] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2425–2433.

[17] D. Teney, L. Liu, and A. v. d. Hengel, "Graph-structured representations for visual question answering," in *CVPR*, 2017.

[18] P. Wang, Q. Wu, C. Shen, and A. van den Hengel, "The vqa-machine: Learning how to use existing vision algorithms to answer new questions," in *CVPR*, 2017.

[19] P. Wang, Q. Wu, C. Shen, A. Dick, and A. van den Hengel, "Fvqa: Fact-based visual question answering," *PAMI*, 2017.

[20] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C Lawrence Zitnick, and R. Girshick, "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2901–2910.

[21] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.

[22] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 3861–3870.

[23] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5736–5745.

[24] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 132–149.

[25] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[27] D. P. Kingma, T. Salimans, and M. Welling, "Improving variational inference with inverse autoregressive flow," *arXiv preprint arXiv:1606.04934*, 2016.

[28] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[29] E. L. Denton, S. Chintala, R. Fergus, *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, 2015, pp. 1486–1494.

[30] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," *arXiv preprint ArXiv:1611.04076*, 2016.

[31] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[32] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," *arXiv preprint arXiv:1912.04958*, 2019.

[33] E. S. Spelke and K. D. Kinzler, "Core knowledge.," *Developmental science*, vol. 10 1, pp. 89–96, 2007.

[34] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[35] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, "Stacked attention networks for image question answering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 21–29.

[36] J. Lu, C. Xiong, D. Parikh, and R. Socher, "Knowing when to look: Adaptive attention via a visual sentinel for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 375–383.

[37] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *CVPR*, vol. 3, 2018, p. 6.

[38] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. A. Forsyth, "Every picture tells a story: Generating sentences from images," in *ECCV*, 2010.

[39] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg, "Baby talk: Understanding and generating simple image descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2891–2903, 2011.

[40] P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi, "Collective generation of natural image descriptions," in *ACL*, 2012.

[41] M. Mitchell, J. Dodge, A. Goyal, K. Yamaguchi, K. Stratos, X. Han, A. Mensch, A. C. Berg, T. L. Berg, and H. Daumé, "Midge: Generating image descriptions from computer vision detections," in *EACL*, 2012.

[42] S. Tellexll, P. Thakerll, R. Deitsl, D. Simeonovl, T. Kollar, and N. Royl, "Toward information theoretic human-robot dialog," *Robotics*, p. 409, 2013.

[43] I. Lütkebohle, J. Peltason, L. Schillingmann, C. Elbrechter, B. Wrede, S. Wachsmuth, and R. Haschke, "The curious robot-structuring interactive robot learning," in *International Conference on Robotics and Automation*, 2009.

[44] D. Skočaj, A. Vrečko, M. Mahnič, M. Janíček, G.-J. M. Kruijff, M. Hanheide, N. Hawes, J. L. Wyatt, T. Keller, K. Zhou, *et al.*, "An integrated system for interactive continuous learning of categorical knowledge," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 28, no. 5, pp. 823–848, 2016.

[45] Y. Yu, A. Eshghi, and O. Lemon, "Learning how to learn: An adaptive dialogue agent for incrementally learning visually grounded word meanings," in *RoboNLP@ACL*, 2017.

[46] J. Thomason, A. Padmakumar, J. Sinapov, J. Hart, P. Stone, and R. J. Mooney, "Opportunistic active learning for grounding natural language descriptions," 2017.

[47] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708.

[48] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2014, pp. 818–833.

[49] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6541–6549.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[51] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.

[52] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017.

[53] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 4, pp. 834–848, 2017.

[54] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1725–1732.

[55] C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann, "Devnet: A deep event network for multimedia event detection and evidence recounting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2568–2577.

[56] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7794–7803.

[57] J. Lin, C. Gan, and S. Han, "Temporal shift module for efficient video understanding," *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[58] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7132–7141.

[59] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua, "Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5659–5667.

[60] J. Fu, J. Liu, H. Tian, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[61] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[62] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[63] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph r-cnn for scene graph generation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 670–685.

[64] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[65] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 764–773.

[66] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[67] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[68] A. Krizhevsky, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[69] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, 2015.

[70] S. Zagoruyko and N. Komodakis, "Wide residual networks," *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

[71] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, Springer, 2014, pp. 740–755.

[72] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision (IJCV)*, vol. 88, no. 2, pp. 303–338, 2010.

[73] J. Yang, J. Lu, D. Batra, and D. Parikh, *A faster pytorch implementation of faster r-cnn*, `https://github.com/jwyang/faster-rcnn.pytorch`, 2017.

[74] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[75] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[76] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[77] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[78] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *ECCV*, 2016.

[79] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[80] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, and D. Batra, "Visual dialog," in *CVPR*, 2017.

[81] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *IJCV*, vol. 123, no. 1, pp. 32–73, 2017.

[82] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *CVPR*, 2017.

[83] D. Parikh, C. L. Zitnick, and T. Chen, "From appearance to context-based recognition: Dense labeling in small images," in *CVPR*, 2008.

[84] A. Oliva and A. Torralba, "The role of context in object recognition," *Trends in cognitive sciences*, vol. 11, no. 12, pp. 520–527, 2007.

[85] L. Ladicky, C. Russell, P. Kohli, and P. H. Torr, "Graph cut based inference with co-occurrence statistics," in *ECCV*, 2010.

[86] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *ICCV*, 2007.

[87] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, "Visual relationship detection with language priors," in *ECCV*, 2016.

[88] B. Zhuang, L. Liu, C. Shen, and I. Reid, "Towards context-aware interaction recognition for visual relationship detection," in *ICCV*, 2017.

[89] J. Peyre, I. Laptev, C. Schmid, and J. Sivic, "Weakly-supervised learning of visual relations," in *ICCV*, 2017.

[90] H. Zhang, Z. Kyaw, J. Yu, and S.-F. Chang, "Ppr-fcn: Weakly supervised visual relation detection via parallel pairwise r-fcn," 2017.

[91] H. Zhang, Z. Kyaw, S.-F. Chang, and T.-S. Chua, "Visual translation embedding network for visual relation detection," in *CVPR*, 2017.

[92] Y. Li, W. Ouyang, and X. Wang, "Vip-cnn: A visual phrase reasoning convolutional neural network for visual relationship detection," in *CVPR*, 2017.

[93] X. Liang, L. Lee, and E. P. Xing, "Deep variation-structured reinforcement learning for visual relationship and attribute detection," in *CVPR*, 2017.

[94] B. Dai, Y. Zhang, and D. Lin, "Detecting visual relationships with deep relational networks," in *CVPR*, 2017.

[95] Y. Li, W. Ouyang, B. Zhou, K. Wang, and X. Wang, "Scene graph generation from objects, phrases and region captions," in *ICCV*, 2017.

[96] A. Newell and J. Deng, "Pixels to graphs by associative embedding," in *NIPS*, 2017.

[97] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, "Neural motifs: Scene graph parsing with global context," in *CVPR*, 2018.

[98] J. Zhang, M. Elhoseiny, S. Cohen, W. Chang, and A. Elgammal, "Relationship proposal networks," in *CVPR*, 2017.

[99] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[100] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *CVPR*, 2018.

[101] X. Gao, B. Xiao, D. Tao, and X. Li, "A survey of graph edit distance," *Pattern Analysis and Applications*, vol. 13, no. 1, pp. 113–129, 2010.

[102] C.-L. Lin, "Hardness of approximating graph transformation problem," in *International Symposium on Algorithms and Computation*, Springer, 1994, pp. 74–82.

[103] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[104] M Everingham, L Van Gool, C. Williams, J Winn, and A Zisserman, "The pascal visual object classes challenge 2012 results," in *See http://www. pascal-network. org/challenges/VOC/voc2012/workshop/index. html*, vol. 5, 2012.

[105] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[106] K. C. Gowda and G Krishna, "Agglomerative clustering using the concept of mutual nearest neighbourhood," *Pattern recognition*, vol. 10, no. 2, pp. 105–112, 1978.

[107] T. Kurita, "An efficient agglomerative clustering algorithm using a heap," *Pattern Recognition*, vol. 24, no. 3, pp. 205–209, 1991.

[108] Y. Gdalyahu, D. Weinshall, and M. Werman, "Self-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1053–1074, 2001.

[109] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.

[110] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[111] G. J. McLachlan and D. Peel, *Finite mixture models*. John Wiley & Sons, 2004.

[112] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, 2002, pp. 849–856.

[113] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Advances in neural information processing systems*, 2005, pp. 1601–1608.

[114] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Departmental Papers (CIS)*, p. 107, 2000.

[115] C. H. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 1, pp. 45–55, 2008.

[116] D. Cai, X. He, X. Wang, H. Bao, and J. Han, "Locality preserving nonnegative matrix factorization," in *IJCAI*, 2009.

[117] S. Zafeiriou and M. Petrou, "Nonlinear non-negative component analysis algorithms," *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 1050–1066, 2009.

[118] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," in *European Conference on Computer Vision*, Springer, 2012, pp. 73–86.

[119] C. Doersch, A. Gupta, and A. A. Efros, "Mid-level visual element discovery as discriminative mode seeking," in *Advances in neural information processing systems*, 2013, pp. 494–502.

[120] B. Hariharan, J. Malik, and D. Ramanan, "Discriminative decorrelation for clustering and classification," in *European Conference on Computer Vision*, Springer, 2012, pp. 459–472.

[121] D. Han and J. Kim, "Unsupervised simultaneous orthogonal basis clustering feature selection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5016–5023.

[122] K. Rematas, B. Fernando, F. Dellaert, and T. Tuytelaars, "Dataset fingerprints: Exploring image collections through data mining," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4867–4875.

[123] H.-C. Huang, Y.-Y. Chuang, and C.-S. Chen, "Affinity aggregation for spectral clustering," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 773–780.

[124] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[125] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[126] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[127] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *2007 IEEE conference on computer vision and pattern recognition*, IEEE, 2007, pp. 1–8.

[128] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[129] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 1096–1103.

[130] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval.," in *ESANN*, vol. 1, 2011, p. 2.

[131] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th annual international conference on machine learning*, ACM, 2009, pp. 609–616.

[132] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *2013 IEEE international conference on acoustics, speech and signal processing*, IEEE, 2013, pp. 8595–8598.

[133] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," in *Advances in neural information processing systems*, 2014, pp. 766–774.

[134] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1422–1430.

[135] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2794–2802.

[136] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[137] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller, "A deep semi-nmf model for learning hidden representations," in *International Conference on Machine Learning*, 2014, pp. 1692–1700.

[138] P. Xie and E. P. Xing, "Integrating image clustering and codebook learning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[139] G. Chen, "Deep learning with nonparametric clustering," *ArXiv*, vol. abs/1501.03084, 2015.

[140] Z. Wang, S. Chang, J. Zhou, M. Wang, and T. S. Huang, "Learning a task-specific deep architecture for clustering," in *Proceedings of the 2016 SIAM International Conference on Data Mining*, SIAM, 2016, pp. 369–377.

[141] J. F. Navarro, C. S. Frenk, and S. D. White, "A universal density profile from hierarchical clustering," *The Astrophysical Journal*, vol. 490, no. 2, p. 493, 1997.

[142] D. Zhao and X. Tang, "Cyclizing clusters via zeta function of a graph," in *Advances in Neural Information Processing Systems*, 2009, pp. 1953–1960.

[143] W. Zhang, X. Wang, D. Zhao, and X. Tang, "Graph degree linkage: Agglomerative clustering on a directed graph," in *European Conference on Computer Vision*, Springer, 2012, pp. 428–441.

[144] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1386–1393.

[145] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[146] W. Zhang, D. Zhao, and X. Wang, "Agglomerative clustering via maximum incremental path integral," *Pattern Recognition*, vol. 46, no. 11, pp. 3056–3065, 2013.

[147] X. Chen and D. Cai, "Large scale spectral clustering with landmark-based representation," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[148] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, ACM, 2003, pp. 267–273.

[149] S. A. Nene, S. K. Nayar, H. Murase, *et al.*, "Columbia object image library (coil-20)," 1996.

[150] D. B. Graham and N. M. Allinson, "Characterising virtual eigensignatures for general purpose face recognition," in *Face Recognition*, Springer, 1998, pp. 446–456.

[151] T. Sim, S. Baker, and M. Bsat, "The cmu pose, illumination, and expression (pie) database," in *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, IEEE, 2002, pp. 53–58.

[152] L. Wolf, T. Hassner, and I. Maoz, *Face recognition in unconstrained videos with matched background similarity*. IEEE, 2011.

[153] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, ACM, 2014, pp. 675–678.

[154] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, IEEE, vol. 2, 2006, pp. 2169–2178.

[155] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[156] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," 2008.

[157] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[158] A. Coates and A. Y. Ng, "Selecting receptive fields in deep networks," in *Advances in neural information processing systems*, 2011, pp. 2528–2536.

[159] Y. Jia, C. Huang, and T. Darrell, "Beyond spatial pyramids: Receptive field learning for pooled image features," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 3370–3377.

[160] T.-H. Lin and H. Kung, "Stable and efficient representation learning with non-negativity constraints," in *International Conference on Machine Learning*, 2014, pp. 1323–1331.

[161] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[162] I. J. Goodfellow, A. Courville, and Y. Bengio, "Spike-and-slab sparse coding for unsupervised feature discovery," *arXiv preprint arXiv:1201.3382*, 2012.

[163] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[164] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in neural information processing systems*, 2005, pp. 513–520.

[165] L. Van Der Maaten, "Learning a parametric embedding by preserving local structure," in *Artificial Intelligence and Statistics*, 2009, pp. 384–391.

[166] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[167] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[168] E. L. Denton, S. Chintala, R. Fergus, *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, 2015, pp. 1486–1494.

[169] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *arXiv preprint arXiv:1606.03498*, 2016.

[170] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *arXiv preprint arXiv:1606.03657*, 2016.

[171] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *European Conference on Computer Vision*, Springer, 2016, pp. 597–613.

[172] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

[173] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, "Generating images with recurrent adversarial networks," *arXiv preprint arXiv:1602.05110*, 2016.

[174] H. Kwak and B.-T. Zhang, "Generating images part by part with composite generative adversarial networks," *arXiv preprint arXiv:1607.05387*, 2016.

[175] E. Mansimov, E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Generating images from captions with attention," *arXiv preprint arXiv:1511.02793*, 2015.

[176] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv preprint arXiv:1605.05396*, 2016.

[177] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, "Learning what and where to draw," *arXiv preprint arXiv:1610.02454*, 2016.

[178] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," *arXiv preprint arXiv:1603.05631*, 2016.

[179] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *arXiv preprint arXiv:1609.02612*, 2016.

[180] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200," California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010.

[181] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[182] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[183] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *International journal of computer vision*, vol. 40, no. 1, pp. 49–70, 2000.

[184] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.046239*, 2015.

[185] S. M. A. Eslami, N. Heess, T. Weber, Y. Tassa, K. Kavukcuoglu, and G. E. Hinton, "Attend, infer, repeat: Fast scene understanding with generative models," *CoRR*, vol. abs/1603.08575, 2016.

[186] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *CoRR*, vol. abs/1601.06759, 2016.

[187] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," *CoRR*, vol. abs/1512.00570, 2015.

[188] P. Isola and C. Liu, "Scene collaging: Analysis and synthesis of natural images with semantic layers," in *IEEE International Conference on Computer Vision*, 2013, pp. 3048–3055.

[189] T. Darrell and A. Pentland, "Robust estimation of a multi-layered motion representation," *IEEE Workshop on Visual Motion*, 1991.

[190] J. Wang and E. Adelson, "Representing moving images with layers," *IEEE Transactions on Image Processing*, 1994.

[191] A. Kannan, N. Jojic, and B. Frey, "Generative model for layers of appearance and deformation," *AISTATS*, 2005.

[192] N. L. Roux, N. Heess, J. Shotton, and J. Winn, "Learning a generative model of images by factoring appearance and shape," *Neural Computation*, vol. 23, pp. 593–650, 2011.

[193] J. Huang and K. Murphy, "Efficient inference in occlusion-aware generative models of images," *CoRR*, vol. abs/1511.06362, 2015.

[194] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 2017–2025.

[195] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, 2007.

[196] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, and D. Batra, "Visual Dialog," 2017.

[197] H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, *et al.*, "From captions to visual concepts and back," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1473–1482.

[198] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4651–4659.

[199] Z. Wu and R. Cohen, "Encode, review, and decode: Reviewer module for caption generation," *arXiv preprint arXiv:1605.07912*, 2016.

[200] J. Lu, C. Xiong, D. Parikh, and R. Socher, "Knowing when to look: Adaptive attention via a visual sentinel for image captioning," 2016.

[201] A. Das, H. Agrawal, C. L. Zitnick, D. Parikh, and D. Batra, "Human attention in visual question answering: Do humans and deep networks look at the same regions?" In *EMNLP*, 2016.

[202] G. Kulkarni, V. Premraj, V. Ordonez, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg, "Babytalk: Understanding and generating simple image descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2891–2903, 2013.

[203] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3128–3137.

[204] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth, "Every picture tells a story: Generating sentences from images," in *ECCV*, Springer, 2010, pp. 15–29.

[205] P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi, "Collective generation of natural image descriptions," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, Association for Computational Linguistics, 2012, pp. 359–368.

[206] M. Mitchell, X. Han, J. Dodge, A. Mensch, A. Goyal, A. Berg, K. Yamaguchi, T. Berg, K. Stratos, and H. Daumé III, "Midge: Generating image descriptions from computer vision detections," in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, 2012, pp. 747–756.

[207] R. Kiros, R. Salakhutdinov, and R. Zemel, "Multimodal neural language models," in *International Conference on Machine Learning*, 2014, pp. 595–603.

[208] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, "Deep captioning with multimodal recurrent neural networks (m-rnn)," *arXiv preprint arXiv:1412.6632*, 2014.

[209] X. Chen and C Lawrence Zitnick, "Mind's eye: A recurrent visual representation for image caption generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2422–2431.

[210] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[211] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[212] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," 2017.

[213] J. Johnson, A. Karpathy, and L. Fei-Fei, "Densecap: Fully convolutional localization networks for dense captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4565–4574.

[214] S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg, "Referitgame: Referring to objects in photographs of natural scenes," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 787–798.

[215] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2641–2649.

[216] R. Hu, M. Rohrbach, J. Andreas, T. Darrell, and K. Saenko, "Modeling relationships in referential expressions with compositional modular networks," pp. 4418–4427, 2017.

[217] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell, "Natural language object retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4555–4564.

[218] A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele, "Grounding of textual phrases in images by reconstruction," in *European Conference on Computer Vision*, Springer, 2016, pp. 817–834.

[219] R. Luo and G. Shakhnarovich, "Comprehension-guided referring expressions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7102–7111.

[220] J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy, "Generation and comprehension of unambiguous object descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 11–20.

[221] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg, "Modeling context in referring expressions," in *European Conference on Computer Vision*, Springer, 2016, pp. 69–85.

[222] L. Anne Hendricks, S. Venugopalan, M. Rohrbach, R. Mooney, K. Saenko, and T. Darrell, "Deep compositional captioning: Describing novel object categories without paired training data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1–10.

[223] S. Venugopalan, L. Anne Hendricks, M. Rohrbach, R. Mooney, T. Darrell, and K. Saenko, "Captioning images with diverse objects," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5753–5761.

[224] T. Yao, Y. Pan, Y. Li, and T. Mei, "Incorporating copying mechanism in image captioning for learning novel objects," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6580–6588.

[225] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Guided open vocabulary image captioning with constrained beam search," *arXiv preprint arXiv:1612.00576*, 2016.

[226] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2692–2700.

[227] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[228] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[229] D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 740–750.

[230] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd an-*

nual meeting of the association for computational linguistics: system demonstrations, 2014, pp. 55–60.

[231] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2002, pp. 311–318.

[232] M. Denkowski and A. Lavie, "Meteor universal: Language specific translation evaluation for any target language," in *Proceedings of the ninth workshop on statistical machine translation*, 2014, pp. 376–380.

[233] R. Vedantam, C Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4566–4575.

[234] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in *ECCV*, Springer, 2016, pp. 382–398.

[235] R. Luo, *Unofficial pytorch implementation for self-critical sequence training for image captioning*, 2017.

[236] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "VQA: Visual Question Answering," in *International Conference on Computer Vision (ICCV)*, 2015.

[237] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei, "Visual7w: Grounded question answering in images," 2016.

[238] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," 2017.

[239] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," 2017.

[240] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. v. d. Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," 2018.

[241] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied Question Answering," 2018.

[242] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "Iqa: Visual question answering in interactive environments," 2018.

[243] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Active learning with gaussian processes for object categorization," 2007.

[244] X. Li and Y. Guo, "Adaptive active learning for image classification," 2013.

[245] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–114, 2012.

[246] S. Vijayanarasimhan and K. Grauman, "Large-scale live active learning: Training object detectors with crawled data and crowds," vol. 108, no. 1-2, pp. 97–114, 2014.

[247] P. Bachman, A. Sordoni, and A. Trischler, "Learning algorithms for active learning," 2017.

[248] G. Contardo, L. Denoyer, and T. Artieres, "A meta-learning approach to one-step active learning," *arXiv preprint arXiv:1706.08334*, 2017.

[249] M. Fang, Y. Li, and T. Cohn, "Learning how to active learn: A deep reinforcement learning approach," 2017.

[250] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Multi-class active learning for image classification," 2009.

[251] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," 2008.

[252] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, Ieee, 2009, pp. 248–255.

[253] I. Misra, R. Girshick, R. Fergus, M. Hebert, A. Gupta, and L. van der Maaten, "Learning by Asking Questions," 2018.

[254] Y. Sun, L. Bo, and D. Fox, "Learning to identify new objects," 2014.

[255] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016.

[256] M. R. Loghmani, B. Caputo, and M. Vincze, "Recognizing objects in-the-wild: Where do we stand?," 2018.

LIST OF PUBLICATIONS

1. **Jianwei Yang**, Zhile Ren, Chuang Gan, Hongyuan Zhu, Devi Parikh. Cross-channel Communication Networks. Advances in Neural Information Processing (**NeurIPS**) 2019.

2. **Jianwei Yang***, Zhile Ren*, Mingze Xu, Xinlei Chen, David Crandall, Devi Parikh, Dhruv Batra. Embodied Amodal Recognition: Learning to Move to Perceive Objects. International Conference on Computer Vision (**ICCV**) 2019.

3. **Jianwei Yang***, Jiasen Lu*, Stefan Lee, Dhruv Batra, Devi Parikh. Visual Curiosity: Learning to Ask Questions to Learn Visual Recognition. Conference on Robotic Learning (**CoRL**) 2018. **Oral Presentation**.

4. **Jianwei Yang***, Jiasen Lu*, Stefan Lee, Dhruv Batra, Devi Parikh. Graph R-CNN for Scene Graph Generation. European conference on computer vision (**ECCV**) 2018.

5. Jiasen Lu*, **Jianwei Yang***, Dhruv Batra, Devi Parikh. Neural Baby Talk. Conference on Computer Vision and Pattern Recognition (**CVPR**) 2018. **Spotlight Presentation**.

6. **Jianwei Yang**, Anitha Kannan, Dhruv Batra, Devi Parikh. LR-GAN: Layered Recursive Generative Adversarial Networks for Image Generation. International Conf. on Learning Representations (**ICLR**), 2017.

7. Jiasen Lu, Anitha Kannan, **Jianwei Yang**, Devi Parikh, Dhruv Batra. Best of Both Worlds: Transferring Knowledge from Discriminative Learning to a Generative Visual Dialog Model. Neural Information Processing Systems (**NIPS**), 2017.

8. Jiasen Lu, **Jianwei Yang**, Dhruv Batra, Devi Parikh. Hierarchical Question-Image Co-Attention for Visual Question Answering. Neural Information Processing Systems (**NIPS**), 2016.

9. **Jianwei Yang**, Devi Parikh, Dhruv Batra. Joint Unsupervised Learning of Deep Representations and Image Clusters. IEEE International Conference on Computer Vision and Pattern Recognition (**CVPR**), 2016.

* Equal Contribution